A DEMONSTRATION DIGITAL COMPUTER

By

D. RADHAKRISHNAN

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY 1975

A DEMONSTRATION DIGITAL COMPUTER

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of

MASTER OF TECHNOLOGY

By
D. RADHAKRISHNAN

to the

DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

JULY 1975

DEDICATED TO

MY TEACHERS WHO TAUGHT ME

THE FUNDAMENTALS OF DIGITAL ELECTRONICS

EE- 1975- M- RAD -DEM

C E R T I F I C A T E

Certified that this thesis entitled 'A Demonstration Digital Computer' has been carried out under my supervision by Mr. D. Radhakrishnan and has not been submitted elsewhere for a degree.

(V. RAJARAMAN)
Professor
Department of Electrical Engineering,
Indian Institute of Technology Kanpur.

July, 1975.

ACKNOWLEDGEMENT

I am deeply indebted to Prof. V. Rajaraman, my thesis supervisor who suggested the problem to me, inspired confidence in me, led me out of unclear mazes wherein I was groping, and helped me in myriad other ways which I cannot even attempt to list. Words do not suffice to express my gratitude towards him.

I am also thankful to Mr. P.C. Mathias of ACES for his timely and ready help during the course of the work. Thanks are also due to the Ministry of Education, Government of India and the authorities of the College of Engineering, Trivandrum for permitting me to join the M.Tech Programme under QIP.

Mr. C.M. Abraham deserves to be commended on the excellent job of typing he did.

And there are the little acts of kindness which everyone of my friends and associates showed to me, they are too numerous to be listed. To these unnamed persons go my heartfelt thanks, and you, dear reader, are not the least of them.

Kanpur,
July, 1975.

D. RADHAKRISHNAN

A B S T R A C T

The design and fabrication of a Demonstration Digital Computer is described in this thesis. Since the accent on cost minimization has been strong in the design, the machine is within the financial capabilities of the Engineering Colleges of the Country. It is expected to be useful as a teaching aid in a digital circuits laboratory, and acquaints the user with the essentials of computer hardware.

The main features of the machine are an 8 bit 32 word memory, manual and automatic modes of operation, binary input/output, simultaneous display of all registers and control flip flops, and ease of fault-location and correction.

A few sample machine-language programs for help to the non-adept user are also listed.

C O N T E N T S

CHAPTER I

INTRODUCTION

Few people see, touch, or use computers, but most people do feel the impact of computer technology on their daily lives. This has been mainly because of the sophistication and wide applicability of the machine. In the last decade attempts have been made to simplify the system design with a view to reducing cost and making the machine accessible to more people. This has led to the development of efficient and economic technological breakthrough like microprocessors. But with equipments of this type the user never gets an insight of the units which control and compute data at submicrosecond rates. A step by step process for control and computation is required for a clear insight into the actual operation. The design of the Demonstration Computer described in this study has been done giving greater importance to this aspect.

Nowadays there is a switch over from analog circuits to digital circuits. This is because of the digital circuits' advantages, their simplicity and noise immunity over the conventional analog circuits. This forced most of the Educational Institutions to offer courses in digital circuits.

One of the most important needs of an Educational institution is well equipped laboratories. For a digital circuits laboratory it is necessary to have a good variety of digital IC chips and at the same time to have some small systems using them to give a good understanding of their applications. A study of these systems helps the student to get a closer acquaintance with the design techniques used with such circuits and also assists him in designing new systems.

The work described in this thesis was taken up with this aspect in mind. A DEMOnstration Computer (DEMOC) is designed and fabricated so that the student can write machine language programs and operate them on the computer. This enables him to get a thorough knowledge of the art of writing programs and at the same time he acquires a good knowledge of the binary and octal number systems.

The system provides two modes of operations - Automatic and Manual. By operating it in the automatic mode he can have a check on his program-whether the sequence is correct or not. This is of help in the development of good algorithms. The manual mode of operation on the other hand helps him to trace the various steps involved in each operation. This gives a clear understanding of the development of microsteps

for new operation codes. This also helps him to have a clear picture of the internal details of a small computer system.

The work involved could have been considerably reduced by making use of several new products currently available in the market such as microprocessors, ROM's etc in the design, but this would have involved sacrificing some of the major objectives of this project. The different steps involved in the execution of each operation code cannot be clearly demonstrated if a microprocessor is used.

With the present design, the components used cost about Rupees Five thousand for the entire system. This reduction in cost has necessitated reduction in storage capacity and limitation on computational capability. Developments in modern device technology like TTL integrated circuits, LED's etc have been exploited to its full extent and great care has been taken to use indegenous components as far as possible. Low cost with added facilities makes it useful in teaching the basics of subjects like Digital Electronics.

Obviously this demonstration computer cannot hope to substitute the elaborate and expensive computers that are in use at present. What it hopes to achieve is to introduce the age of computers to Educational institutions which cannot otherwise even dream of acquiring a computer because of paucity of funds.

CHAPTER II

SYSTEM DESIGN

This chapter discusses the design considerations and choice of machine features in detail. It also includes a detailed description of system organisation, functional units, word format, instruction list and system specifications. Operating instructions and some simple programs are given at the end of the chapter.

2.1 Machine features:

The following factors are of great importance in choosing the basic machine features.

2.1.1 Serial or parallel operation:

(a) Circuitry:

(i) Cost:-In a parallel machine addition is performed simultaneously for all the bits. Hence separate adder circuit is essential for each bit. Serial machine on the other hand uses one adder and addition is carried out bit by bit. The circuit cost difference between these two types of machines increases with word length. Hence parallel machines are more expensive.

(ii) Complexity of control: The timing sequence is simple in a parallel machine because the information is transferred with in the different units of the computer by the application of a single pulse through several gates. In a serial machine a series of timing pulses corresponding to individual bit positions must be generated and applied to the various gates.

(b) Speed:

Parallel machines are faster compared to serial ones. In parallel machines the basic unit of time is chosen on the basis of addition of two numbers. In serial machines the transfer of word from one unit to another determines the basic clock period.

(c) Compatibility with type of control:

Serial machine is compatible with synchronous control only whereas parallel machines can have either synchronous or asynchronous control.

2.1.2 Synchronous or Asynchronous operation:

Operation in the digital computer is in time sequence; control of this sequence can be either synchronous or asynchro ous. In a synchronous computer, each logic operation takes place under the control of the clock in synchronism with the

clock pulses. In addition the execution of an instruction such as addition or shift occurs at a fixed time interval, usually the longest required for various operations. In an asynchronous computer, no fixed time reference exists for the operations. Completion of any one operation produces an operation complete signal; this signal initiates the next operation. An asynchronous computer will be faster because it does not have to 'use up' a fixed time interval. If most of the operations of the computer can be arranged to perform in one or more constant time intervals, then there is not much gain in asynchronous operation.

In a synchronous system only limited tolerance is allowed because of requirements on wave shapes and pulse coincidence at gates. Asynchronous systems are operable with components of considerable tolerance.

Most of the computers are synchronous in nature and this may be ascribed to several factors including the following:

(i) There are potential hazards in an asynchronous network due to variation in response of active elements and in transmission time of signals.

(ii) Hazard free operation in asynchronous systems are considered more difficult to design and service.

2.1.3 Numerical representation in the Arithmetic unit:

(a) Number base:

Arithmetic operations are more complex in BCD representation. Binary arithmetic is much simpler to implement.

(b) Radix point:

In fixed-point number systems, the radix point is assumed always to be at the same location with respect to all digit groups - at the left for fractions, at the right for integers, and at some intermediate point for mixed numbers. The floating point number system, by contrast, treats numbers as a fractional part f and an exponent part e, the combination being regarded as representing numbers x such that

$$x = f \cdot r^e$$

where r is the radix of the number system.

The exponent e is regarded as an integer and may be either + or − . The great advantage of the floating point representation is the latitude it affords the programmer in scaling variables. But the floating point representation sacrifices precision to gain range. The arithmetic unit becomes more complex due to the flexibility in its representation of numbers.

(c) Representation of negative numbers:

In the binary representation of a number usually the most significant digit is devoted for indicating its sign. The other digits of a signed binary number are called here the number digits. There are three ways of representing these numbers.

(i) Signed - magnitude representation:

Multiplication and division is easy and input output data conversion is simpler. Addition and subtraction are more complex. There are two values of zero. Separate circuitry must be provided to determine the sign of different operations.

(ii) Two's complement representation :-

Exactly one cycle is required for addition and subtraction of any 2's complement number. Obtaining a complement is easy. Extra hardware is needed for multiplication and division. Overflow detection is simpler and representation of zero is unique.

(iii) One's complement representation :

Addition and subtraction requires an extra cycle for end around carry. This slows down the computer speed. Overflow detection is difficult in comparison to 2's complement representation and representation of zero is not unique.

On the basis of the above discussion on different design factors, the following machine features are chosen for the proposed Demonstration Digital Computer.

(i) Binary fixed point machine

(ii) Parallel operation

(iii) Synchronous control

(iv) One's complement arithmetic

(v) Single address machine

2.2 Word format:

The word length depends on the coded representation of a number as well as an instruction. It is convenient and economical from the point of view of storage to represent both instruction and data by the same number of bits. The word length depends on the following factors:

(i) The number of bits used for op code to distinguish one instruction from another . If there are n different built in instructions, at least $\log_2(n+x)$ bits are required where x is the smallest integer that makes (n+x) an integral power of two.

(ii) The capacity of main storage. The number of bits needed for address selection is $\log_2 N$, where N is the memory capacity.

With the available size of memory storage (16 x 4 solid state RAM) a word length of 8 bits is chosen for DEMOC.

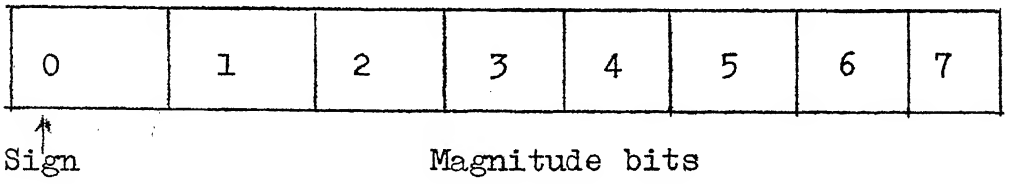
Number of bits used for operation code = 3

Number of bits used for address selection = 5

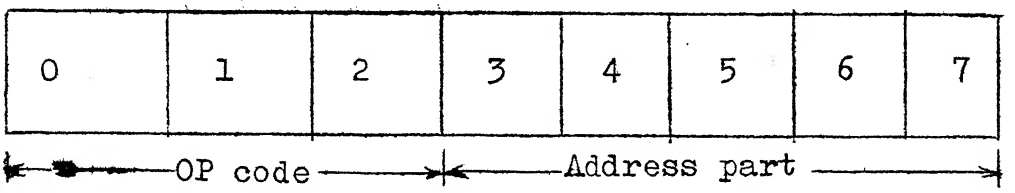
The format of the instructions as well as data are depicted in Figure 2.1.

2.3 Number and type of instructions

A list of all the instructionsis given in Table 2.1. Octal representation is used. While converting to octal form op code and address part are treated separately.



(a) Data format



(b) Instruction format

Fig. 2.1 WORD FORMAT

Table 2.1

INSTRUCTIONS

Octal Code	Mnemonic Symbol	Operation
0	HLT	Stop
1	CLA m	Load m to the accumulator
2	NAND m	NAND accumulator and m and store in accumulator
3	STO m	Contents of accumulator is stored in m.
4	ADD m	Add m to the accumulator.
5 0 4	NOT	Magnitude bits of accumulator is complemented.
5 1 0	SHL	Accumulator shifted one bit to the left.
5 2 0	SHR	Accumulator shifted one bit to the right.
6	BR m	Transfer control to storage location m.
7	BRM m	Transfer control to storage location m if accumulator is negative.

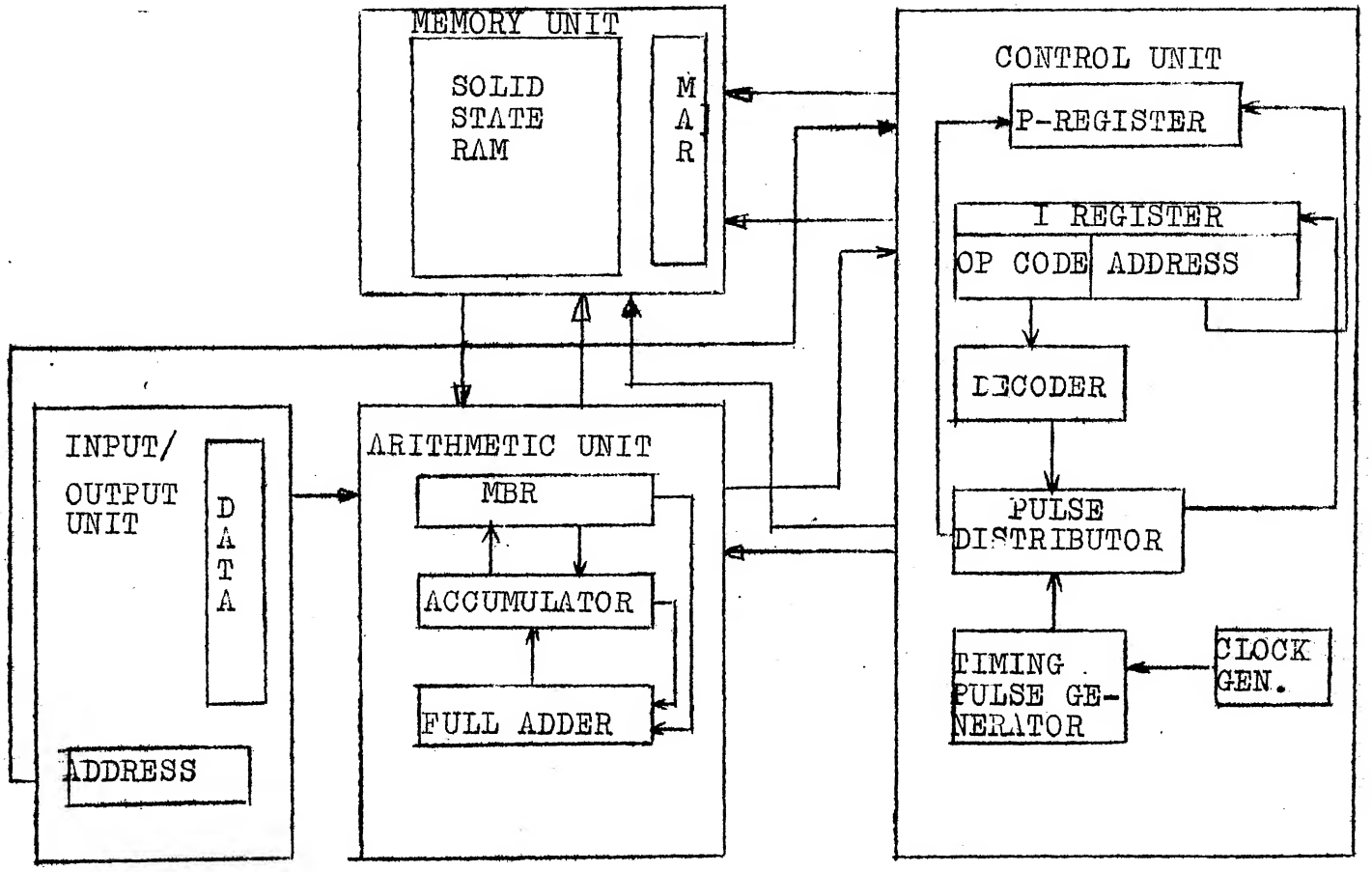


Fig. 2.2 SYSTEM BLOCK DIAGRAM

2.4 System organisation:

A system block diagram showing major functional blocks is given in Fig. 2.2.

Memory address register (MAR) is included in the memory unit. Memory Buffer Register (MBR) is shown external to the unit and is used as the second operand for arithmetic operations. All arithmetic and logical operations are performed in the accumulator. P-Register holds the address of the next instruction to be executed. Instruction register (I) holds the current instruction which is in its execution cycle. Input is given through a set of eight toggle switches. Program and data has to be manually entered into memory through these switches.

A simple clock generator provides all the control signals needed for the system. These control signals are derived from a pulse distributor. The OP code decoder outputs as well as the clock signal are fed to this pulse distributor for producing the various control signals.

Once the program and data are stored into the memory the operation of the machine alternates between the Fetch sequence and Execution sequence. During the Fetchsequence, the instruction to be executed is read from memory and brought to the instruction register. This also clears all the

conditional flip flops and increments p register by unity.

During the execution sequence the instruction in I-Reg is executed depending on the control signals received from the Pulse distributor. Once the execution part is completed the Fetch sequence is again initiated.

2.5 Functional Description:

2.5.1 Memory unit:

The memory storage consists of 8 bit, 32 word RAM. Four solid state memory chips each of 4 bit, 16 word capacity are combined together in a series parallel fashion to make up the above module. The information transmission between the memory unit and the other units of the computer is through MBR.

(i) Memory Buffer Register (MBR): It is an 8 bit register formed by combining two 4 bit shift registers in series.

(ii) Memory Address Register (MAR): Since each memory chip consists of 16 words of 4 bits, a total of 4 wires is sufficient to select any memory location. MAR and its decoding logic are provided inside the chip. This register is controlled by the P-Register or I-Register depending on whether the instruction is in Fetch phase or Execution phase.

2.5.2 Input-Output unit:

There is no separate I/O unit provided in the system. Input is given through a set of 8 toggle switches (DATA) and using two press buttons 'ENTER MBR' and 'WRITE'.

The output of any memory location can be read into the MBR by the 'READ' button. The output is displayed in signed-magnitude form.

2.5.3 Arithmetic unit:

Two four bit full adders form the basis for the arithmetic unit. It works in conjunction with MBR and accumulator. All transfer operations between ACC and MBR are done in parallel mode.

Accumulator:

It is an 8 bit register formed of two 4 bit shift registers. A number of operations are possible in this register, for example, shift left by one bit, shift right by one bit, complement operation etc.

2.5.4 Control unit:

The control unit consists of an instruction register which accepts instruction words from MBR, and control circuits which sequence, translate and execute these instructions by means of a set of sub commands.

Sequencing is accomplished jointly by the memory and control units in a basic machine cycle. The control unit is provided with a counter (P-Reg) which is always set to the address of the next instruction to be executed.

Translation is performed by the decoding circuit and the distributor. It generates the actual electrical signals for the execution of each of the instructions from the I-Reg. and develops a predetermined sequence to transfer the information with in the arithmetic unit and other units of the computer.

Execution depends on timing pulses as well as sub-commands. The control must produce as many sequences of gating or switching signals as the number of different arithmetic, logical and transfer operations the computer is required to perform.

Manual control is available to allow the operator to intervene, monitor, or modify the automatic operation of the computer.

P-Register: It is a 5 bit register and contains the address of the memory location which specifies the next instruction to be executed. Information enters into P-Reg.

from I-Reg. during the execution of branch instructions. Its contents can be incremented by one using the 'INC-P' switch on console.

Instruction mode:

Fetch state: Upon completion of an instruction, the instruction mode flip-flop is set to FETCH state. It allows the address stored in the P-Reg to choose the next instruction word from memory to be transferred to I-Reg. It also increments P-Reg by one. As soon as the instruction is brought to I Reg, the next pulse resets this mode flip-flop to EXECUTE state.

Execute state: In this state, I-Reg transfers address to MAR and Op code to the decoder. The decoder circuit along with the distributor decodes the Op code into sub-commands which execute the instruction. After execution of the instruction, the instruction mode flip flop is set to FETCH state to start the new cycle.

Instruction Register (I-Reg):

This 8 bit register is also formed of two 4 bit shift registers. The first three bits of this register forms the op code and the remaining 5 bits forms the address part for the instruction which is currently in its execution state.

2.6 Console

The front panel of the machine is shown in Figure 2.3. All the registers used in the system are displayed here using LED's. All important control flip flops as well as the control counter are also displayed. In addition to these a testing panel is also provided on the front of the panel. This includes some important test points and also the accumulator output terminals. The latter enable the accumulator output to be displayed on a CRO, if it is so desired.

The 'START' button is for the automatic starting of the program once the mode switch is in the 'AUTO' position. The 'STEP' switch has to be pressed each time for the execution of one microinstruction in 'Manual' mode.

2.7 Operating procedure:

1. Switch on the power supply
2. Select mode switch to 'LOAD' position.
3. Press 'G-CLEAR' button and make sure that the panel indicators 'F' is ON and 'E', 'H', 'OV' and 'COUNTER' are all OFF.
4. Press 'P-CLEAR' and ensure that P-register is cleared by noting the P-register bits displayed on the panel.

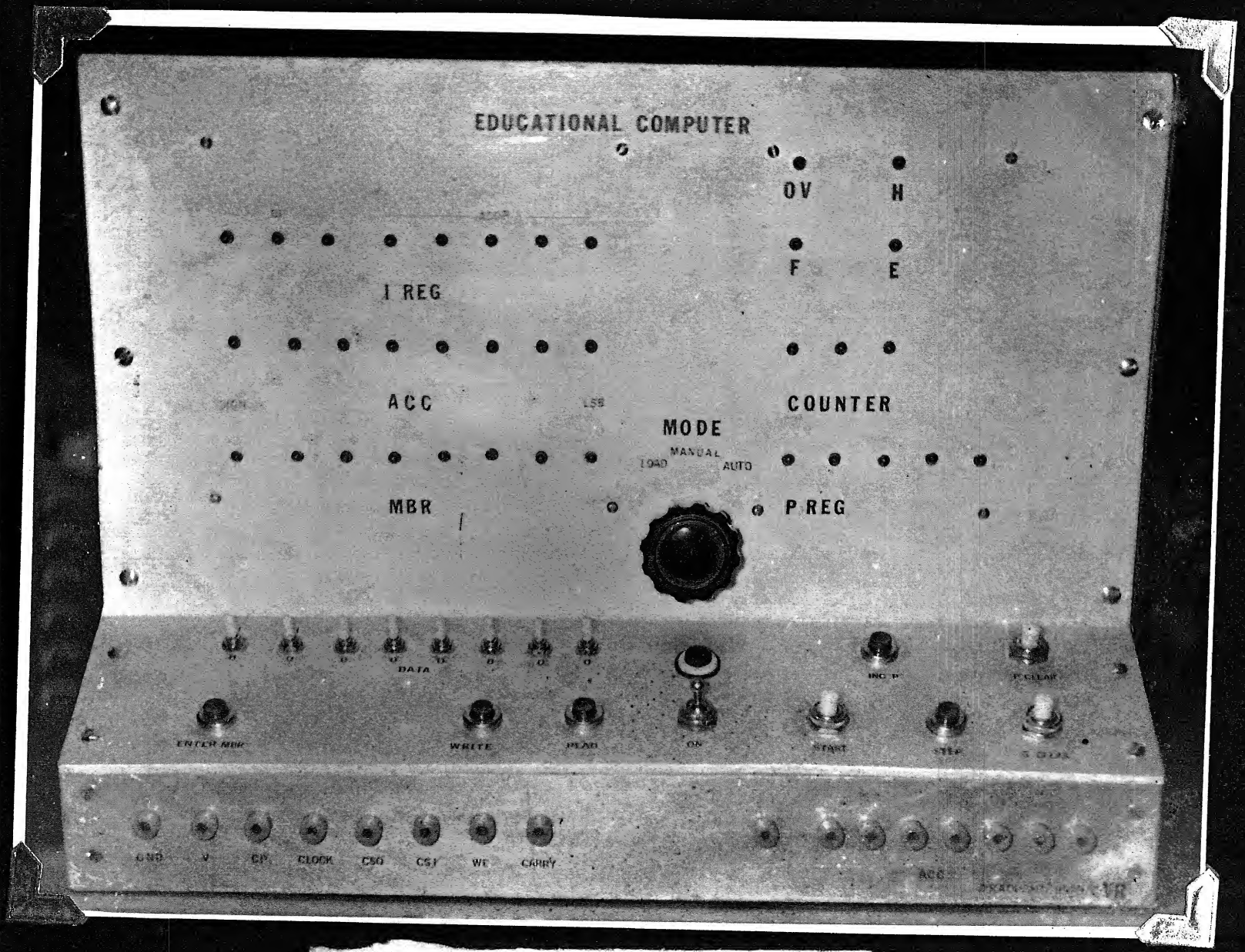


Fig. 2.3 Console Panel

5. Select the input data/instruction on the 'DATA' switches and press 'ENTER MBR' push button. The contents of these switches is now displayed by the MBR register on the panel.
6. Select the appropriate address of memory, by 'INC-P' micro-switch and press 'WRITE' button. This operation writes the contents of MBR into memory.
7. Press 'READ' button once and check for the contents of MBR. If the contents of MBR remains unchanged then go back to step 5 and select the next instruction/data until the complete program is stored. Once this is over then continue from step 8. If the contents of MBR has changed then repeat from step 5 keeping the 'DATA' switches undisturbed.
8. Now for the execution of the program we can select any one of the modes AUTO/MANUAL. If 'AUTO' mode is selected then proceed to step 9; otherwise to step 10.
9. Put the mode switch to 'AUTO' position and press 'START' button. When the computer comes to a halt, 'H' indicator lights up. Now read the contents of the memory location, where result is stored, by selecting the particular address using 'INC-P' and 'READ' button.
10. Put the mode switch to 'MANUAL' mode. Press 'STEP' switch a number of times until 'H' indicator lights up. Now read the contents as given in Step 9.

2.8 Illustrative programs:

A few programs will be of help to the user to get familiarized with the art of writing programs in machine language. A short account of the method of developing these programs is also given before the program per-se.

Program to convert from Binary to Gray Code:

Let the binary number be A. Shift right A by one bit, then the Exclusive OR operation between A and the shifted number gives the Gray Code representation of A. Binary number must be stored in location 15. Result is stored in location 14 at the end of the program.

Location	Instruction	Location	Instruction
0	1 15	10	2 16
1	5 20	11	2 14
2	3 16	12	3 14
3	5 04	13	0 00
4	2 15	14	DATA
5	3 14	15	A
6	1 15		
7	5 04		

Program for sub-traction

X = A - B.

Program changes the sign of B and adds to A. Minuend is stored in location 11 and subtrahend in location 10. Final result is stored in address 6.

Location	0 Instruction	
0	1	10
1	2	07
2	5	04
3	4	11
4	3	06
5	0	00
	DATA	
6	0	
7	7	37
10		B (Subtrahend)
11		A (Minuend)

Program for Multiplication of two numbers

This program makes use of the successive shifted addition depending on the most significant digit of the multiplier. The two numbers to be multiplied are to be stored in location 27 and 30. The contents of locations 25,26 and 30 must be initialized every time the program is being run. The result is stored in location 25.

Location	Instruction	Location	Instruction
0	1 26	15	6 00
1	5 10	16	1 25
2	3 26	17	4 27
3	1 30	20	3 25
4	5 10	21	1 26
5	3 30	22	7 24
6	7 16	23	6 11
7	1 26	24	0 00
			DATA
10	7 24	25	0 00
11	1 25	26	0 01
12	5 10	27	A (Multiplicand)
13	3 25	30	B (Multiplier)
14	7 24		

Program for division of two numbers

X = A/B

This is done by repeated subtraction of the divisor from the dividend. Only the integral part of the result is stored in location 35. The dividend and divisor are stored in locations 36 and 32 respectively. The contents of locations 35,36 and 37 must be initialized at the beginning of each run.

Location	Instruction		Location	Instruction	
0	1	32	21	5	10
1	2	33	22	3	37
2	5	04	23	7	25
3	3	31	24	6	14
4	1	31	25	1	35
5	4	36	26	4	34
6	3	36	27	3	35
7	7	14	30	0	00
				DATA	
10	1	35	31		
11	4	34	32	B	
12	3	35	33	7	37
13	6	04	34	0	01
14	1	36	35	0	00
15	5	10	36	A	
16	3	36	37	0	01
17	7	30			
20	1	37			

Program to convert from Binary to BCD

The maximum BCD number which can be represented is 79. Hence if the binary number exceeds this value then result stored in location 27 will be zero. The binary number to be converted is stored in location 30. The contents of locations 27 and 30 must be initialized at the beginning of each run. 10 is subtracted each time to get the MSD of the BCD number.

Location	Instruction		Location	Instruction	
0	1	30	15	5	10
1	4	26	16	5	10
2	7	04	17	5	10
3	0	00	20	5	10
4	1	30	21	4	30
5	4	25	22	3	27
6	7	14	23	0	00
				DATA	
7	3	30	24	0	01
10	1	27	25	4	12(-10)
11	4	24	26	6	20(-80)
12	3	27	27	0	00
13	6	04	30		A
14	1	27			

Program to convert from BCD to Binary

MSD of the BCD number stored in location 26 is separated and multiplied by 10 by repeated shift and add method and then LSD is added to it. Result in pure binary form is stored in location 23. The contents of locations 23 and 26 must be initialized at the beginning of each run.

Location	Instruction		Location	Instruction	
0	1	26	14	1	26
1	2	22	15	2	24
2	5	04	16	5	04
3	3	26	17	4	23
4	2	25	20	3	23
5	5	04	21	0	00
				DATA	
6	5	20	22	7	37
7	3	23	23		
10	5	20	24	4	17
11	5	20	25	7	20
12	4	23	26	A	
13	3	23			

Program to find the square of a number

The method is based on the fact that the sums of the successive odd integers are equal to the squares of the successive integers.

$$N^2 = 1 + \sum_{i=1}^{N-1} X_{1i}$$

where N is the number to be squared and $X_{1i} = 1 + 2i$

The number is stored in address 23. Contents of addresses 20,21 and 22 must be initialized each time the program is being run. The result is stored in address 20.

Location	Instruction		Location	Instruction	
0	1	23	12	4	22
1	4	22	13	3	22
2	7	15	14	6	00
3	1	21	15	0	00
4	4	20	16	Data 4	01(-1)
5	3	20	17	0	02(2)
6	1	21	20	0	00
7	4	17	21	0	01
10	3	21	22	4	00(-0)
11	1	16	23	N	

Program to find the NOR function of two numbers

$$X = \overline{A + B} = \overline{A} \cdot \overline{B} = \overline{\overline{A} \cdot \overline{B}}$$

The two numbers are stored in locations 13 and 14. Result is stored in location 11. Initialize location 14 before the program is run.

Location	Instruction
0	1 14
1	2 12
2	3 14
3	1 13
4	2 12
5	2 14
6	2 12
7	3 11
10	0 00
	DATA
11	
12	7 37
13	A
14	B

Program to find the Hamming distance between two codes

A bit by bit comparison is made btween the two codes and a count is made if there is disparity between them. The two coded numbers are stored in locations 33 and 34. Result is stored in location 30 at the end of the program. Locations 30, 32, 33 and 34 must be initialized at the beginning of each run.

Location	Instruction		Location	Instruction	
0	1	34	17	1	33
1	7	15	20	7	06
2	5	10	21	5	10
3	3	34	22	3	33
4	1	33	23	1	30
5	7	21	24	4	31
6	5	10	25	3	30
7	3	33	26	6	10
10	1	32	27	0 DATA	00
11	5	10	30	0	00
12	3	32	31	0	01
13	7	27	32	0	01
14	6	00	33	A	
15	5	10	34	B	
16	3	34			

Program for Even parity checking

It is assumed that the parity bit corresponding to the coded word is stored in the LSD (bit 7). If the count of 1's in the coded word (including parity bit) is even then parity is correct. The coded word is stored in location 37. Result comes to location 34. If the result is positive then parity is correct. The contents of locations 34, 35, 36 and 37 must be initialized at the beginning of each run.

Location	Instruction		Location	Instruction	
0	1	37	20	5	10
1	7	30	21	3	34
2	1	37	22	1	35
3	5	10	23	5	10
4	3	37	24	3	35
5	7	13	25	7	27
6	1	36	26	6	17
7	5	10	27	0	00
10	3	36	30	1	34
11	7	17	31	4	35
12	6	02	32	3	34
13	1	34	33	6 DATA	02
14	4	35	34	0	00
15	3	34	35	0	01
16	6	06	36	0	01
17	1	34	37	A	

CHAPTER III

LOGICAL DESIGN

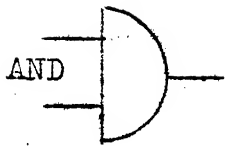
3.1 Logical symbols and diagrams:

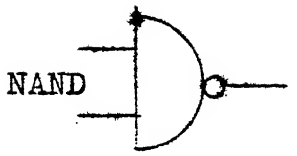
Logical design of the computer is best carried out using the methods of Boolean algebra. Each wire in our logical diagrams is in one of two states (0,1) at each discrete moment of time. The two states 0 and 1 are complementary and they correspond to the truth values: false and true. The different logical symbols and diagrams for building blocks are depicted in Figure 3.1.

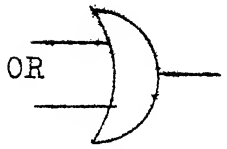
3.2 Choice of Logic family:

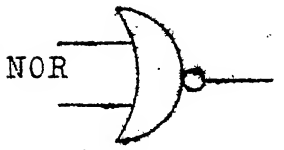
Although the various logic types are identical as to functions performed, logic families can be distinguished by how they perform these functions. An exhaustive comparison centres around the following characteristics.

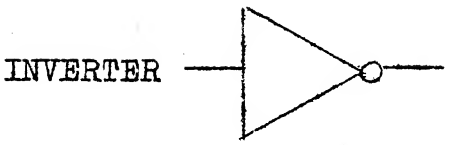
(i) Speed (Propagation time delay)
(ii) Noise immunity
(iii) Fan-in and fan-out capabilities
(iv) Power supply requirements
(v) Power dissipation per gate
(vi) Suitability for integrated fabrication
(vii) Operating temperature range
(viii) Number of functions available
(ix) Cost

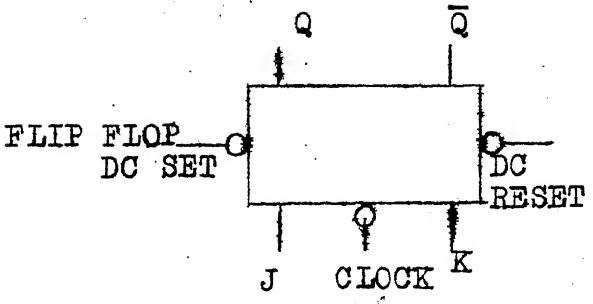












AND	$f \cdot g$	f AND g
OR	$f+g$	f OR g
NEGATION	$\overline{f}$	NOT f
NAND	$\overline{f \cdot g}$	NOT (f AND g)
NOR	$\overline{f+g}$	NOT (f OR g)

Fig. 3.1 Logic Symbols and Diagrams

It should be clear that no single logic configuration can be best suited for all applications. A comparison of the major integrated-circuit digital logic families is given in Table 3.1.

TTL family is chosen for the building blocks of DEMOC and positive logic is used in its design.

3.3 Instruction codes and their control wires:

3.3.1 Design language:

A language, useful in describing digital systems is given here. Alphabets and grammar forms the two major constituents of any language. Digital building blocks such as registers, switches and decoders correspond to variable names in this language. Information transfer from a register to another is the fundamental operation with digital systems.

(i) Declaration: This defines the various components of the digital system .

(a) Register, A(0 - 15), B(1 - 3).

The above statement defines two registers A and B. The 16 bits in A register are numbered as 0,1 ..... 15; from left to right. Similarly, the B register has 3 bits and numbered from 1 to 3.

Table 3.1 Comparison of the major IC digital logic families

Parameter Logic	RTL	DTL	HTL	TTL	ECL	MOS	CMOS
Basic gate	NOR	NAND	NAND	NAND	OR-NOR	NAND	NOR or NAND
Fan out	5	8	10	10	25	20	>50
Power dissipated per gate, mW	12	8-12	55	12-22	40-55	0.2-10	0.01 static 1 at 1 MHz
Noise immunity	Nominal	Good	Excellent	Very good	Good	Nominal	Very good
Propagation delay per gate, ns	12	30	90	12-6	4-1	300	70
Number of functions	High	Fairly high	Nominal	Very High	High	Low	Low
Clock rate, MHz	8	12-30	4	15-60	60-400	2	5

(b) Sub-register, Addr (1 - 6) = I(10 - 15) defines a new register, 'Addr' with six bits and the bits 10 to 15 of the I register form these six bits.

(c) Switch, A ⟨ON, OFF⟩ , MPS ⟨PE, M, A⟩ defines two switches in which A has two possible states and MPS has three positions.

(d) CLOCK, P.

This specifies the name of a clock. When a pulse occurs P becomes 1.

(e) Memory, MEM (00-99) ⟨0:15⟩ defines a memory unit with 100 words and 16 bits/word. The words are addressed from 00 to 99 and bits from 0 to 15.

(ii) Micro operations:

There are four main types of micro operations in digital systems:

(a) Transfer type operations:

These are indicated by a left arrow. |CS| B ← opr A, where 'opr' stands for the operation specified, and CS stands for the control signal.

(b) Memory type micro operations:

They are two in number 'READ' and 'WRITE'.

READ : $GPR_0 \leftarrow M$ (addr)

WRITE : M (addr) $\leftarrow GPR_2$

The address of the memory location selected is enclosed with in parenthesis.

(c) Boolean micro operations:

When A and B are two registers

$C \leftarrow A \cdot B$, would mean

$C(1) \leftarrow A(1) \cdot B(1)$, $C(2) \leftarrow A(2) \cdot B(2)$ and so on. All these bit level operations are done simultaneously.

(d) Functional micro operations:

They are either shift type or count type operations.

ShL 2A shift the contents of register A left twice.

CtP A count up register A by one.

Using the above language the declaration for the proposed DEMOC is as follows:

Memory, MEM $(0 - 37_8)$ <0:7>

Register, ACC (0 - 7), P (0 - 4), MBR (0 - 7),
I (0 - 7), OV, H, F, CARRY

Sub register, OPR(1 - 3) = I(0 - 2),
Addr(1 - 5) = I(3 - 7),
Sr = ACC(0), Sm = MBR(0).

Switches, DATA(0 - 7) ⟨ON, OFF⟩ , INC-P ⟨ON, OFF⟩ ,
MODE ⟨LOAD, MANUAL, AUTO⟩ , ENTER MBR ⟨ON, OFF⟩,
WRITE ⟨ON, OFF⟩ , READ ⟨ON, OFF⟩ ,
START ⟨ON, OFF⟩ , STEP ⟨ON, OFF⟩ ,
P-CLEAR ⟨ON, OFF⟩, G-CLEAR ⟨ON, OFF⟩

Light OV ⟨ON, OFF⟩ , H ⟨ON, OFF⟩ , F ⟨ON, OFF⟩ ,
E ⟨ON, OFF⟩ , COUNTER (0 - 2) ⟨ON, OFF⟩ ,
P(0 - 4) ⟨ON, OFF⟩ , I(0 - 7) ⟨ON, OFF⟩ ,
ACC(0 - 7) ⟨ON, OFF⟩ , MBR(0 - 7) ⟨ON, OFF⟩ .

3.3.2 Instructions and sub-commands:

The different instructions together with their micro-instructions and timing signals are given in this section.

Timing pulse		Operation
	HALT	OP Code 0
$0T_1$	$\lvert D_0 \cdot T_1 \rvert$	F ← 1, H ← 1, Ctr ← 0
	Load	OP Code 1 m
$1T_1$	$\lvert D_1 \cdot T_1 \rvert$	MBR ← M(m) ;
$1T_2$	$\lvert D_1 \cdot T_2 \rvert$	ACC ← MBR ;
$1T_3$	$\lvert D_1 \cdot T_3 \rvert$	F ← 1 , Ctr ← 0
	NAND	Op code 2 m
$2T_1$	$\lvert D_2 \cdot T_1 \rvert$	MBR ← M(m) ;
$2T_2$	$\lvert D_2 \cdot T_2 \rvert$	ACC ← $\overline{ACC.MBR}$;
$2T_3$	$\lvert D_2 \cdot T_3 \rvert$	F ← 1, Ctr ← 0

Timing pulse		Operation
Transfer		Op code 3 m
$3T_1$	$D_3 \cdot T_1$	MBR ← ACC ;
$3T_2$	$D_3 \cdot T_2$	M(m) ← MBR ;
$3T_3$	$D_3 \cdot T_3$	F←1, Ctr←0
Add		Op code 4 m
$4T_1$	$D_4 \cdot T_1$	MBR←M(m) ;
$4T_2$	$D_4 \cdot T_2 \cdot \overline{S_r} \cdot S_m$	MBR (1:7)←$\overline{\text{MBR (1:7)}}$;
$4\overline{T_2}$	$D_4 \cdot T_2 \cdot S_r \cdot \overline{S_m}$	ACC (1:7)←$\overline{\text{ACC(1:7)}}$;
$4T_3$	$D_4 \cdot T_3$	ACC(1:7)←ACC(1:7)+MBR(1:7);
$4\overline{T_3}$	$D_4 \cdot T_3 \cdot C_o$	CARRY ← 1 ;
$4T_4$	$D_4 \cdot T_4 \cdot (S_r \oplus S_m) \cdot \text{CARRY}$	S_r←0, ACC←ACC + 1 ;
$4\overline{T_4}$	$D_4 \cdot T_4 \cdot (S_r \oplus S_m) \cdot \overline{\text{CARRY}}$	S_r←1, ACC ⟨1:7⟩←$\overline{\text{ACC⟨1:7⟩}}$
$4\overline{\overline{T_4}}$	$D_4 \cdot T_4 \; \overline{(S_r \oplus S_m)} \cdot \text{CARRY}$	OV←1 ;
$4\overline{T_5}$	$D_4 \cdot T_5 \cdot \overline{OV}$	F←1, Ctr←0
$4T_5$	$D_4 \cdot T_5 \cdot OV$	F←1, Ctr←0, H←1
Invert		Op Code 5 0 4
504 T_1	$D_{504} \cdot T_1$	ACC ⟨1:7⟩←$\overline{\text{ACC ⟨1:7⟩}}$;
504 T_2	$D_{504} \cdot T_2$	F←1 , Ctr←0
Shift left		Op Code 5 1 0
510 T_1	$D_{510} \cdot T_1$	ACC ⟨0:6⟩←ACC ⟨1:7⟩ ACC⟨7⟩←0;
510 T_2	$D_{510} \cdot T_2$	F←1 , Ctr←0

Timing pulse		operation
	Shift Right	Op code 520
520 T_1	$\lvert D_{520} \cdot T_1 \rvert$	ACC $\langle 1:7 \rangle \leftarrow$ ACC $\langle 0:6 \rangle$, ACC $\langle 0 \rangle \leftarrow 0$;
520 T_2	$\lvert D_{520} \cdot T_2 \rvert$	F $\leftarrow$ 1 Ctr $\leftarrow$ 0
	Unconditional Branch op code 6 m	
6 T_1	$\lvert D_6 \cdot T_1 \rvert$	P $\leftarrow$ m ;
6 T_2	$\lvert D_6 \cdot T_2 \rvert$	F $\leftarrow$ 1, Ctr $\leftarrow$ 0
	Branch on minus op code 7 m	
7 T_1	$\lvert D_7 \cdot S_r \cdot T_1 \rvert$	P $\leftarrow$ m;
7 T_2	$\lvert D_7 \cdot T_2 \rvert$	F $\leftarrow$ 1, Ctr $\leftarrow$ 0

S_r - Sign bit of Accumulator

S_m - Sign bit of MBR

C_o - End around carry during addition

3.3.3 Sequence of Events:

The general sequence of events in the operation of DEMOC is listed below using the microinstructions. The Fetch cycle is common for all instructions.

1) $\lvert \bar{H} \cdot F \cdot C_p \rvert$ OV $\leftarrow$ 0, CARRY $\leftarrow$ 0, MBR $\leftarrow$ M(P) ;

2) $\lvert \bar{H} \cdot F \cdot C_p \rvert$ I $\leftarrow$ MBR, CTP $\leftarrow$ P ;

3) $\lvert \bar{H} \cdot F \cdot C_p \rvert$ F $\leftarrow$ 0, COUNTER $\leftarrow$ 0

4) What OP Code?

(a) HLT $H \leftarrow 1$,

(b) CLA m $MBR \leftarrow M(m)$

$ACC \leftarrow MBR$ GO TO 1

(c) NAND m $MBR \leftarrow M(m)$

$ACC \leftarrow \overline{ACC \cdot MBR}$ GO TO 1

(d) STO m $MBR \leftarrow ACC$

$M(m) \leftarrow MBR$ GO TO 1

(e) ADD m $MBR \leftarrow M(m)$

$ACC \leftarrow ACC + MBR$ if O'flow lights then $H \leftarrow 1$

(f) NOT $ACC(1\text{-}7) \leftarrow \overline{ACC(1\text{-}7)}$ GO TO 1

(g) SHL $ACC(0\text{-}6) \leftarrow ACC(1\text{-}7)$, $ACC(7) \leftarrow 0$ GO TO 1

(h) SHR $ACC(1\text{-}7) \leftarrow ACC(0\text{-}6)$, $ACC(0) \leftarrow 0$ GO TO 1

(i) BR m $P(0\text{-}4) \leftarrow Addr(1\text{-}5)$ GO TO 1

(j) BRM m $P(0\text{-}4) \leftarrow Addr(1\text{-}5)$ if ACC is negative, GO TO 1

Steps 1,2 and 3 are referred to as FETCH state, and 4 as EXECUTE state.

3.3.4 Basic micro instructions:

If we go through the sequence of commands, we see that some of the commands are repeated in different instructions. Basic micro instructions used in these instructions along with their timing signals are given in Table 3.2.

Table 3.2

Micro instructions	Timing Signal
1. M(MAR)←MBR	$3T_2$ + WRITE
2. MBR← DATA	ENTER MBR (Manual)
3. MBR←M(MAR)	$1T_1+2T_1+4T_1+t_1$+READ
4. MBR←ACC	$3T_1$
5. MBR(0)← MBR(0)	$4T_2$
6. MBR(1-7)←$\overline{\text{MBR(1-7)}}$	$4T_2$
7. ACC← MBR	$1T_2$
8. ACC(0)← ACC(0)	$504T_1+4\overline{T_2}+4T_3$
9. ACC(1-7)←$\overline{\text{ACC(1-7)}}$	$504T_1+4\overline{T_2}+4\overline{T_4}$
10. ACC←$\overline{\text{ACC}\cdot\text{MBR}}$	$2T_2$
11. ACC(1-7)←ACC(1-7)+MBR(1-7)	$4T_3$
12. ACC(0)← 0	$520T_1+4T_4$
13. ACC(7)←0	$510T_1$
14. ACC(0-6)←ACC(1-7)	$510T_1$
15. ACC(1-7)←ACC(0-6)	$520T_1$
16. CTP ACC	$4T_4$
17. P(0-4)←Addr(1-5)	$6T_1+7T_1$
18. I←MBR	t_2
19. CTP P	t_2 + INC-P
20. ACC(0)←1	$4\overline{T_4}$

Micro instructions	Timing Signal
21. F ← 0	t_3
22. F ← 1	$0T_1+(1+2+3)T_3+D_5 \cdot T_2+4T_5+6T_2+7T_2+G\text{-}CLEAR$
23. H ← 0	G-CLEAR
24. H ← 1	$0T_1 + 4T_5$
25. CARRY ← 1	$4\overline{T_3}$
26. CARRY ← 0	G-CLEAR + t_1
27. COUNTER ← 0	$0T_1+1T_3+2T_3+3T_3+D_5 \cdot T_2+4T_5+6T_2+7T_2+t_3+G\text{-}CLEAR$
28. OV ← 0	t_1 + G-CLEAR
29. OV ← 1	$4\overline{\overline{T_4}}$
30. P ← 0	P-CLEAR

The signals ENTER MBR, WRITE, READ, INC-P, G-CLEAR and P-CLEAR are external pulses applied manually using switches. t_1, t_2 and t_3 denotes the 1st, 2nd and 3rd pulse during the Fetch sequence. The entire logic design can be carried out using these basic microinstructions.

3.3.5 ISP Descriptive Systems:

The ISP (Instruction-set Processor) is meant to provide a uniform way of describing instruction sets, that is, to give the information contained in a programming manual. It must provide the instruction format, the registers referenced by the instructions, the rules of interpretation of the instruction, and the semantics of each instruction in the processor's repertoire. An ISP level of description for DEMOC is given in Appendix 1.

3.4 Memory unit:

3.4.1 Memory Control Logic:

The memory (32 x 8) is divided into two sets of 16 words each. Each set consists of one pair of memory chips to form a 16 x 8 module. Each memory chip has two control inputs - Chip Select (CS) and Write Enable (WE) The purpose of these two inputs is given below.

CS = 0 Read/Write

WE = 0 Write

The CS inputs for the two sets are denoted as CS_0 and CS_1. CS_0 enables the set consisting of words from 0 to 15 and CS_1 enables the set consisting of words from 16 to 31. The logic equations for these control signals are given below:

$$\overline{CS_0} = C_p \cdot READ \cdot \overline{P_0} + C_p \cdot WRITE \cdot \overline{P_0} + t_1 \cdot \overline{P_0} + \overline{I_3} \cdot (1T_1 + 2T_1 + 3T_2 + 4T_1)$$

On simplification we get

$$CS_0 = \overline{(C_p \cdot READ) \cdot \overline{P_0} + (C_p \cdot WRITE) \cdot \overline{P_0} \cdot t_1 \cdot \overline{P_0} + \overline{I_3} \cdot T_{11}}$$

where $T_{11} = 1T_1 + 2T_1 + 3T_2 + 4T_1$, Similarly

$$CS_1 = \overline{(C_p \cdot READ) \cdot P_0 + (C_p \cdot WRITE) \cdot P_0 \cdot \overline{t_1} \cdot P_0 + I_3 \cdot T_{11}}$$

$\overline{WE} = D3 \cdot T_2 + C_p \cdot WRITE$ on simplification

$WE = \overline{D3 \cdot T_2} \cdot \overline{C_p \cdot WRITE}$

The logic circuit is developed in Figure 3.2.

All AND functions are realized by NAND gates due to the non-availability of AND gates.

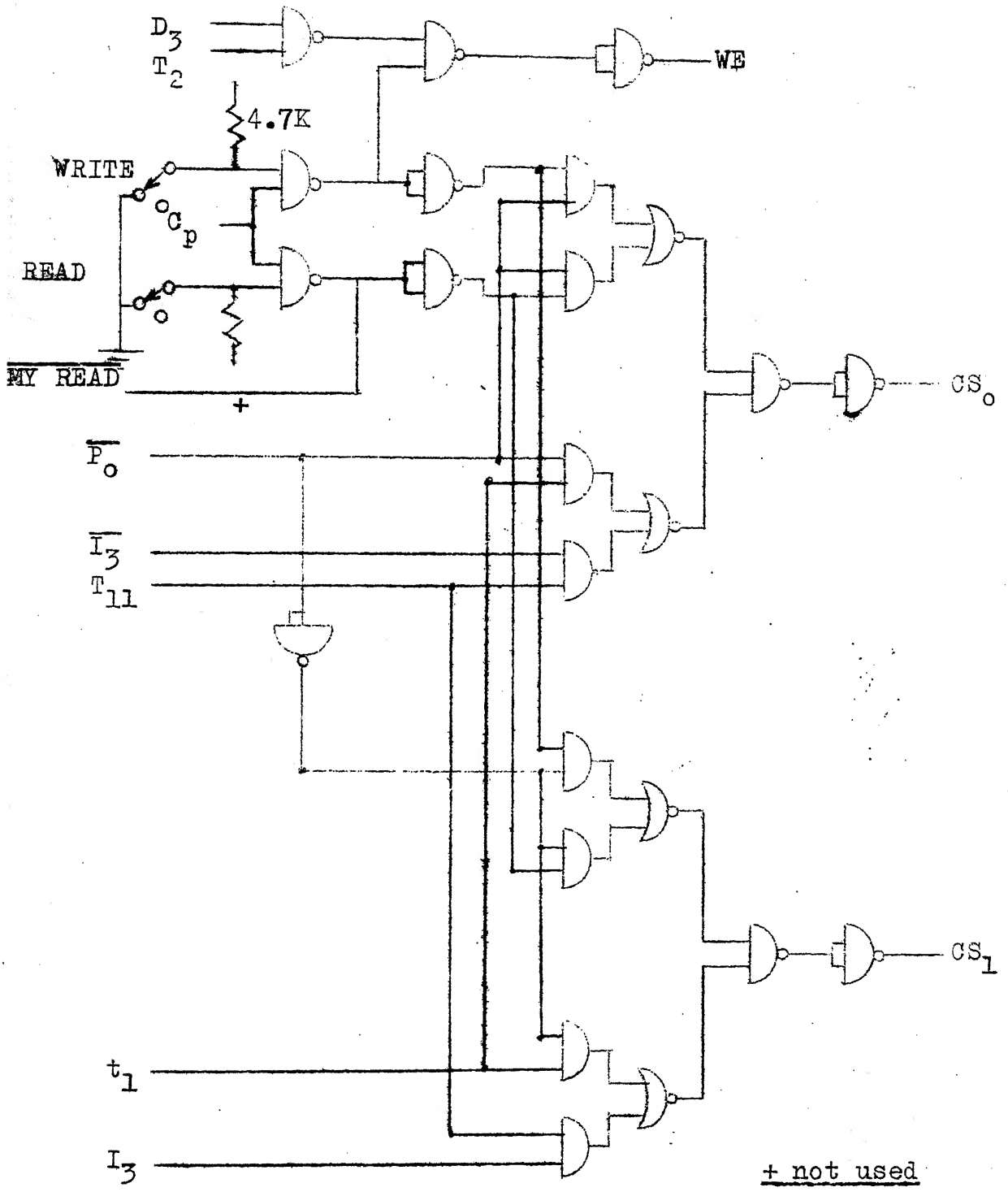


Fig. 3.2 Memory Control Logic.

3.4.2 Memory address selection:

The memory capacity of 32 words necessitates 5 bit registers for address selection. But the memory chip provides only 4 address lines at its input. Hence only the least significant 4 bits of the P-register and I register are used for address selection. The address selection logic is

$$a_0 = F \cdot P_1 + \bar{F} \cdot I_4 = \overline{\overline{F \cdot P_1} \cdot \overline{\bar{F} \cdot I_4}}$$

$$a_1 = F \cdot P_2 + \bar{F} \cdot I_5 = \overline{\overline{F \cdot P_2} \cdot \overline{\bar{F} \cdot I_5}}$$

$$a_2 = F \cdot P_3 + \bar{F} \cdot I_6 = \overline{\overline{F \cdot P_3} \cdot \overline{\bar{F} \cdot I_6}}$$

$$a_3 = F \cdot P_4 + \bar{F} \cdot I_7 = \overline{\overline{F \cdot P_4} \cdot \overline{\bar{F} \cdot I_7}}$$

The logic circuitry is given in Figure 3.3.

3.4.3 Data input and output:

Data enters into memory from MBR. Similarly data output from memory goes to MBR. Since all memory chips provide open collector outputs in complement form, the NOR function is realized easily by tying them together and returning to power supply through a resistor as shown in Figure 3.4.

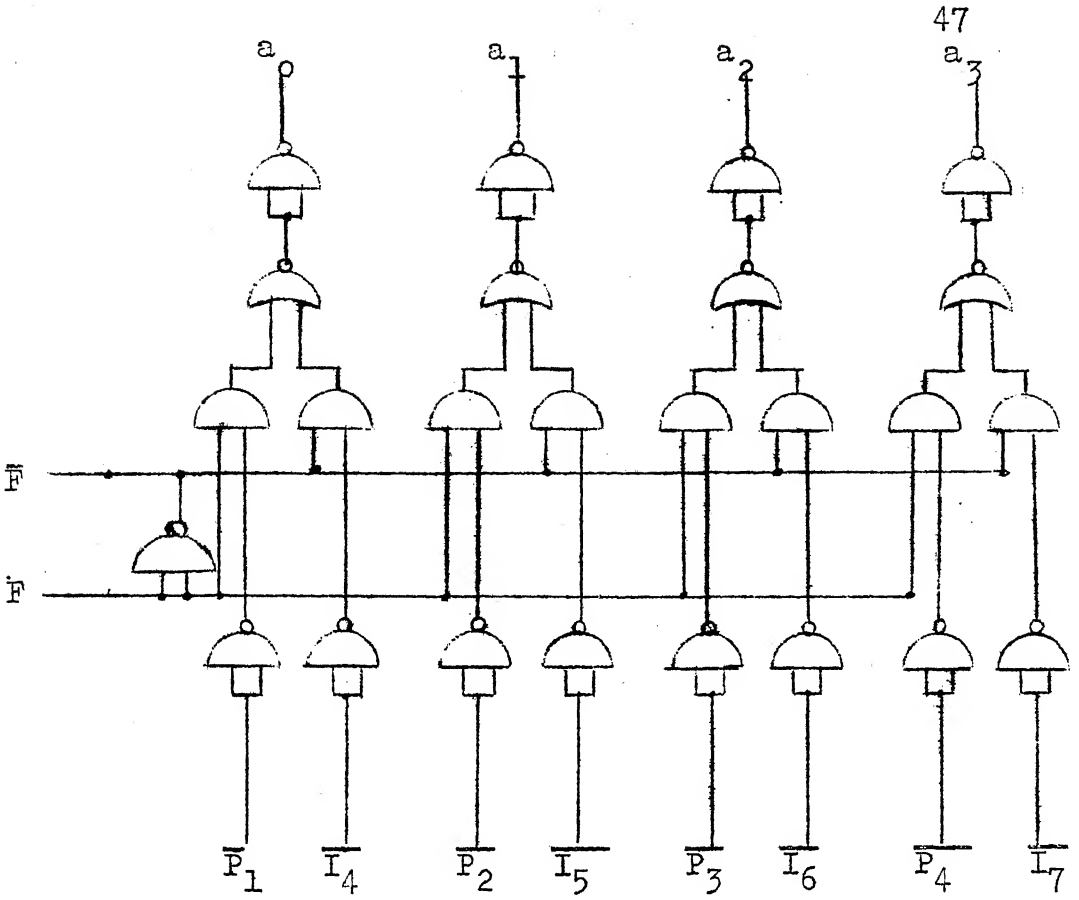


Fig. 3.3 Memory address selection

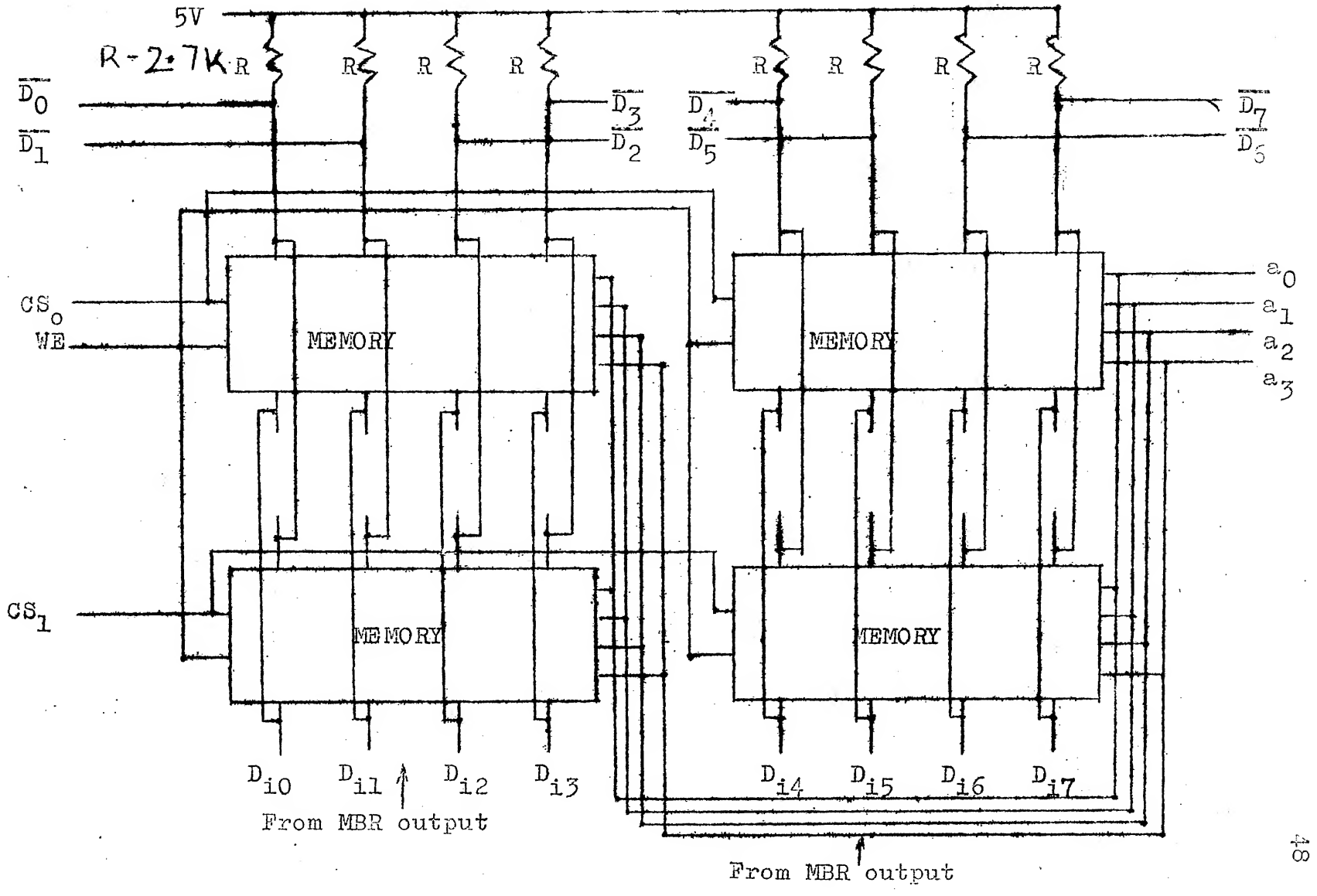


Fig. 3.4 Memory data input and output

3.4.4 Memory Buffer Register:

The output of these memory chips goes to MBR. MBR is controlled by the mode control input and clock 2 input in its parallel transfer mode of operation. In this mode of operation 'mode control' input must be kept at '1' logic level. The logic equations for these two inputs are given as

$$\text{Mode Control} = 1T_1+2T_1+3T_1+4T_1+4T_2+t_1+C_p\cdot\text{ENTER MBR}+C_p\cdot\text{READ}$$

$$= \overline{\overline{C_p\cdot\text{ENTER MBR}\cdot 3T_1}\cdot\overline{4T_2}+t_1\cdot\overline{1T_1+2T_1}\cdot(\overline{4T_1}\cdot\overline{C_p\cdot\text{READ}})}$$

$$\text{CLOCK 2} = \text{CLOCK} + C_p\cdot\text{READ}+C_p\cdot\text{ENTER MBR}$$

$$= \overline{\overline{\text{CLOCK}}\cdot\overline{C_p\cdot\text{READ}}\cdot\overline{C_p\cdot\text{ENTER MBR}}}$$

The logic circuitry for these two inputs are developed in Figure 3.5.

The information enters into MBR from four sources-

(i) From Accumulator

(ii) From Memory unit

(iii) From Data input switches and

(iv) From MBR itself.

The logic equations for the MBR inputs thus becomes :

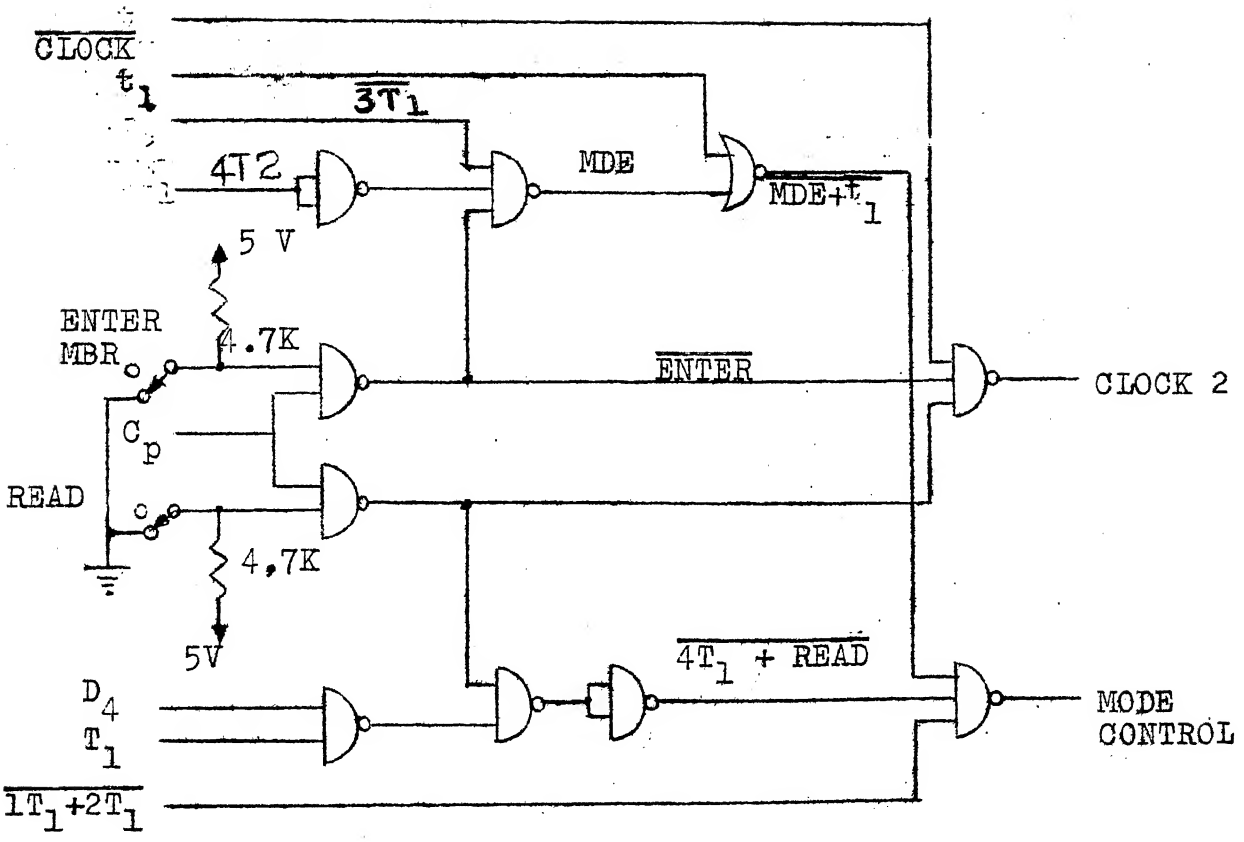


Fig. 3.5 MBR Control logic

Input 0 = $D_0 + M_{i0} + 4T_2 \cdot M_0 + 3T_1 \cdot ACC(0)$

$$= \overline{\overline{D_0} \cdot \overline{M_{i0}} \cdot \overline{4T_2 \cdot M_0} + 3T_1 \cdot ACC(0)}$$

Inputs 1 through 7 are similar.

Input 1 = $D_1 + M_{i1} + 4T_2 \cdot \overline{M_1} + 3T_1 \cdot ACC(1)$

and realized as $\overline{\overline{D_1} \cdot \overline{M_{i1}} \cdot \overline{4T_2 \cdot M_1} + 3T_1 \cdot ACC(1)}$

where $\overline{D_0}$, $\overline{D_1}$ ---- $\overline{D_7}$ comes directly as the outputs of memory chips and $\overline{M_{i0}}$, $\overline{M_{i1}}$ ---$\overline{M_{i7}}$ are the outputs of the DATA switches.

The complete logic circuitry is shown in Figure 3.6.

3.5 Input-Output unit:

The information selected in the 'DATA' input switches enters MBR when 'ENTER MBR' is activated. The logic circuit for this is developed in Figure 3.7.

Results are stored in memory. Hence the output can be read into MBR by the read procedure mentioned earlier.

3.6 Arithmetic unit:

Addition is the only arithmetic operation provided in DEMOC. Addition is carried out between accumulator and MBR. In addition to this some shift operations as well as complementation are possible in accumulator.

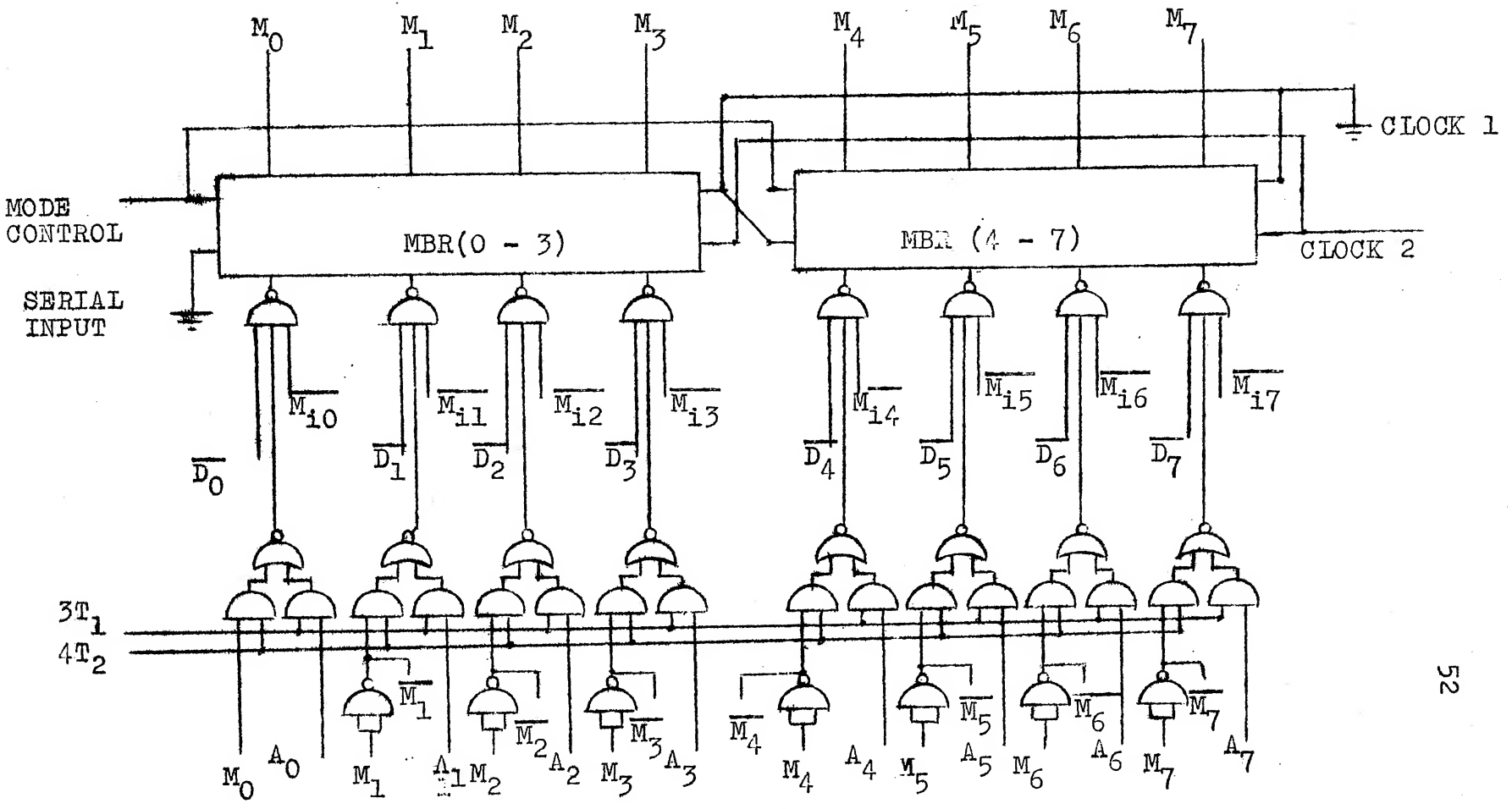


Fig. 3.6 MBR input circuitry

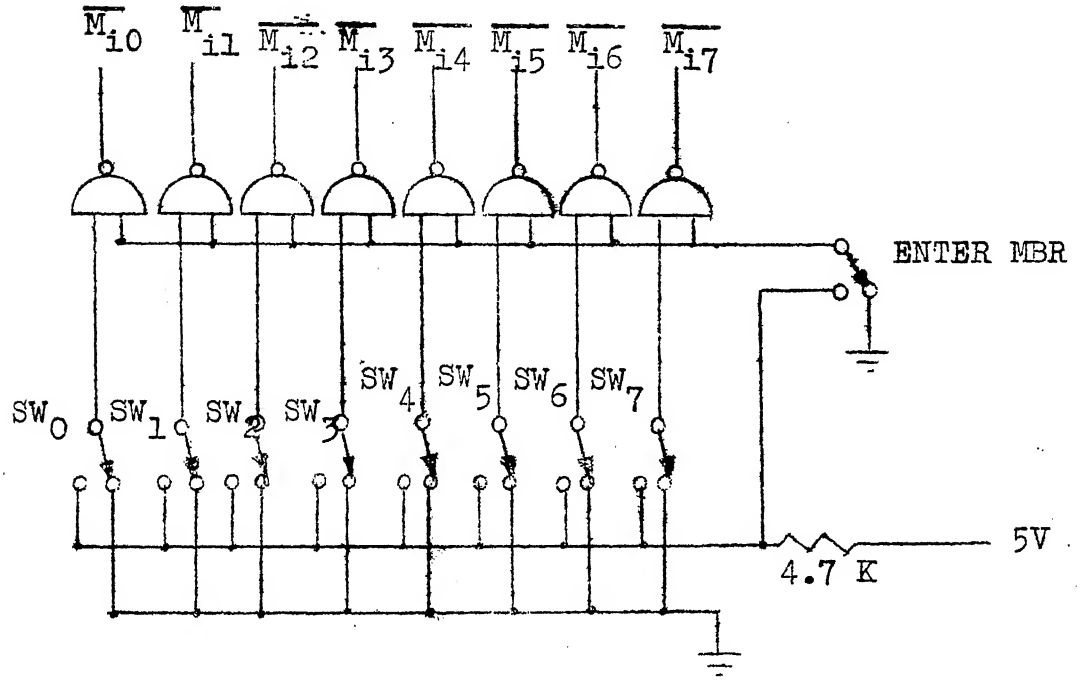


Fig. 3.7 DATA input

3.6.1 Binary full Adder:

Two four bit binary full adders are used in the circuit. Since we need only 7 bit addition the most significant digit inputs are permananently connected to '0'. The sum output corresponding to this bit comes out as the end around carry C_0.

3.6.2 Accumulator:

Accumulator consists of a pair of 4 bit shift registers.

(i) Design of control circuit: The logical equations for the clock and control inputs are given:

$$\text{CLOCK 1} = 520T_1 \cdot \text{CLOCK}$$

$$= \overline{\overline{520T_1} + \overline{\text{CLOCK}}}$$

$$\text{CLOCK 2} = \text{CLOCK} \cdot (1T_2 + 2T_2 + 4T_3 + 510T_1 + \text{INVRT} + 4T_4)$$

$$= \overline{\overline{1T_2} \cdot \overline{2T_2} \cdot \overline{4T_3} \cdot \overline{510T_1} \cdot \overline{\text{INVRT}} \cdot \overline{4T_4} + \overline{\text{CLOCK}}}$$

where $\text{INVRT} = 504T_1 + 4\overline{T_2} + 4\overline{T_4}$

$\text{MODE CONTROL} = \overline{520T_1}$

(ii) Logic for data input: The logic for the different inputs is derived here. The CTP ACC logic follows from the usual

Boolean simplification as:

$$A_{i7} = 4T_4 \cdot \overline{A_7}$$

$$A_{i6} = 4T_4 \cdot (A_6 \cdot \overline{A_7} + \overline{A_6} \cdot A_7)$$

$$A_{i5} = 4T_4 \cdot (\overline{A_5} \cdot A_6 \cdot A_7 + A_5 \cdot \overline{A_6} + A_5 \cdot \overline{A_7})$$

$$A_{i4} = 4T_4 \cdot (\overline{A_4} \cdot A_5 \cdot A_6 \cdot A_7 + A_4 \cdot \overline{A_5} + A_4 \cdot \overline{A_6} + A_4 \cdot \overline{A_7})$$

$$A_{i3} = 4T_4 \cdot (\overline{A_3} \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 + A_3 \cdot \overline{A_4} + A_3 \cdot \overline{A_5} + A_3 \cdot \overline{A_6} + A_3 \cdot \overline{A_7})$$

$$= 4T_4 \cdot (\overline{A_3} \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 + A_3 \cdot \overline{(A_4 \cdot A_5 \cdot A_6 \cdot A_7)})$$

$$A_{i2} = 4T_4 \cdot (\overline{A_2} \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 + A_2 \cdot \overline{A_3} + A_2 \cdot \overline{A_4 \cdot A_5 \cdot A_6 \cdot A_7})$$

$$A_{i1} = 4T_4 \cdot (\overline{A_1} \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 + A_1 \cdot \overline{A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7}$$

$$= 4T_4 \cdot (\overline{A_1} \cdot A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7 + \overline{\overline{A_1} + A_2 \cdot A_3 \cdot A_4 \cdot A_5 \cdot A_6 \cdot A_7})$$

where A_{i1}, A_{i2}, ---------A_{i7} are the inputs to the accumulator. Logic for all other operations follows very easily and all these are combined together. The complete circuit diagram is given in Figures 3.8 and 3.9.

3.7 Control unit:

The control unit consists of a decade counter and its associated circuitry, instruction register, OP code decoder, a pulse distributor unit, a few control FLIP-FLOPS, P-register, and a clock pulse generator. A detailed logic design of each

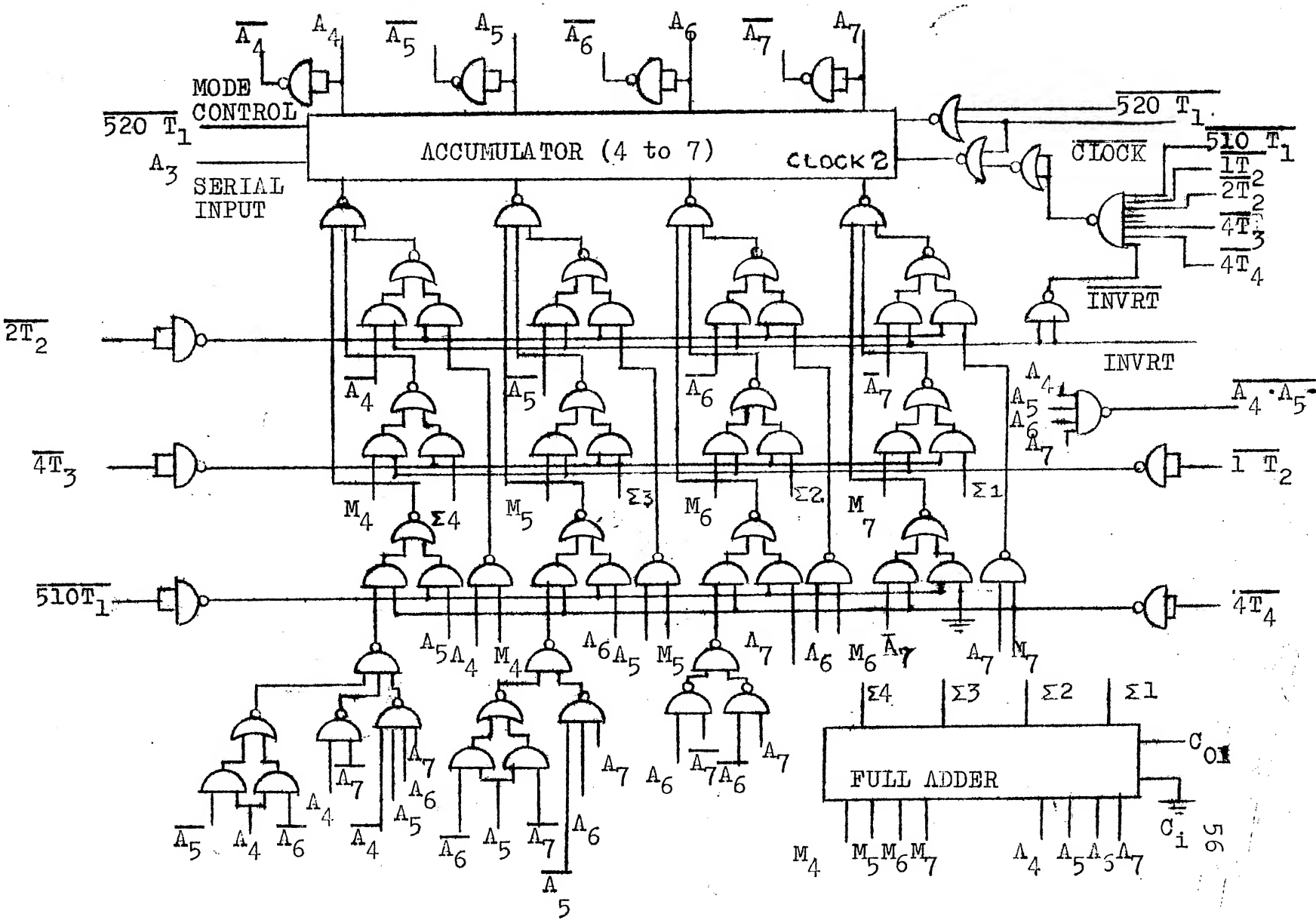


Fig. 3.8 Accumulator bits 4 to 7

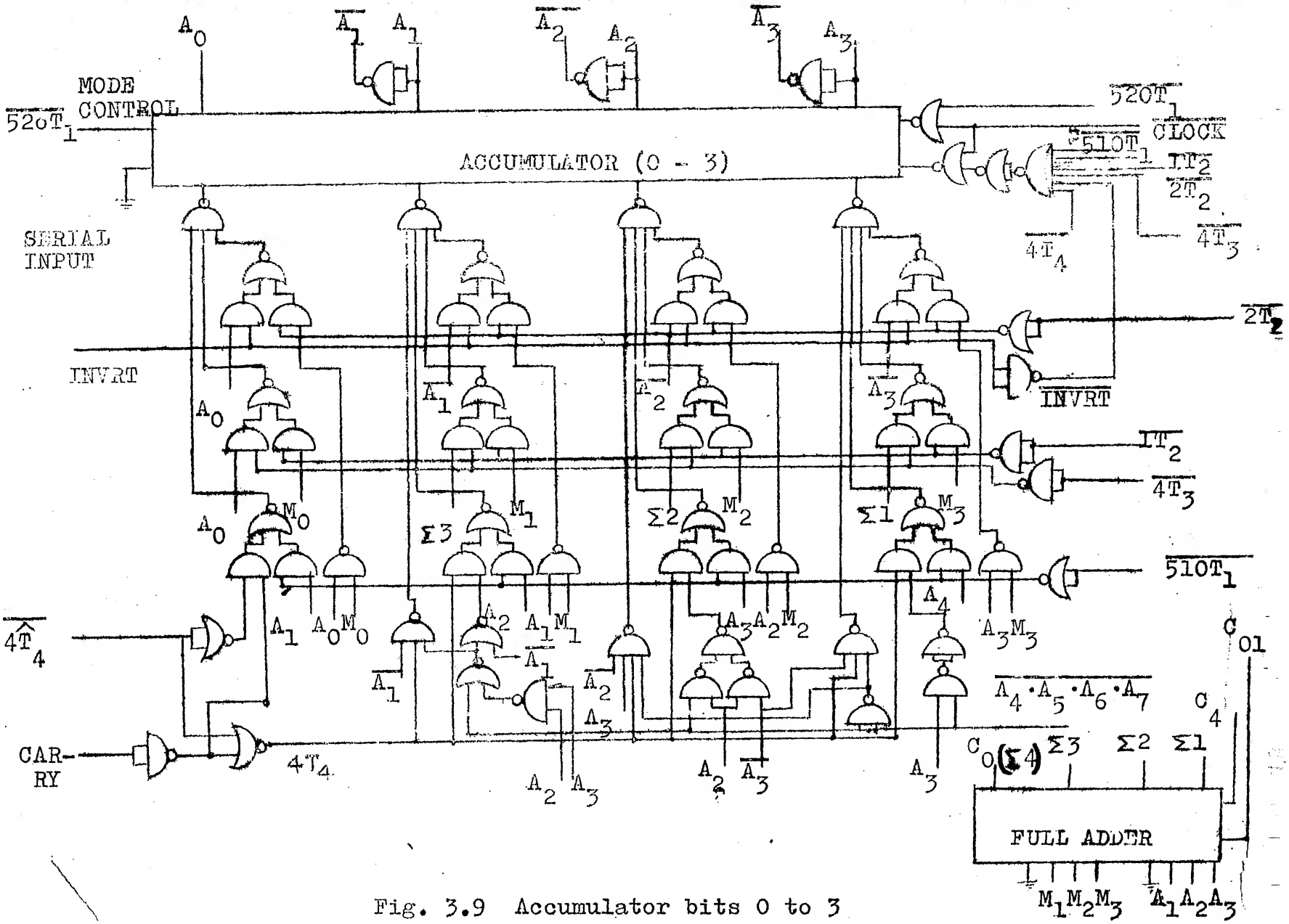


Fig. 3.9 Accumulator bits 0 to 3

of these circuits follows:

3.7.1 Decade counter:

This is a four bit binary counter, which produces the timing pulses for the execution as well as the fetch sequences. The clock input to this counter is controlled by the 'HALT', and 'FETCH' FLIP-FLOPS . At the end of each cycle this counter is reset to zero. Figure 3.10 shows this counter and its associated circuitry.

3.7.2 Control FLIP-FLOPS:

(a) HALT FLIP-FLOP (H): This FLIP-FLOP when it is set blocks the entrance of clock pulses to the counter. This gets set when the Halt instruction is executed or when an overflow occurs during the addition operation. The G-clear pulse resets this to zero state.

(b) FETCH FLIP-FLOP: (F): The setting and resetting of this FLIP-FLOP is controlled by the execution and fetch pulses. The end of execution pulse T_{31} sets this to the Fetch state and the end of Fetch pulse t_3 resets it to the Execution state. This FLIP-FLOP is initialized to the Fetch state by the G-clear pulse.

(c) CARRY FLIP-FLOP (CARRY): This FLIP-FLOP stores the end around carry during addition operation. This FLIP-FLOP is reset to zero in every Fetch cycle as well as by the G-clear pulse.

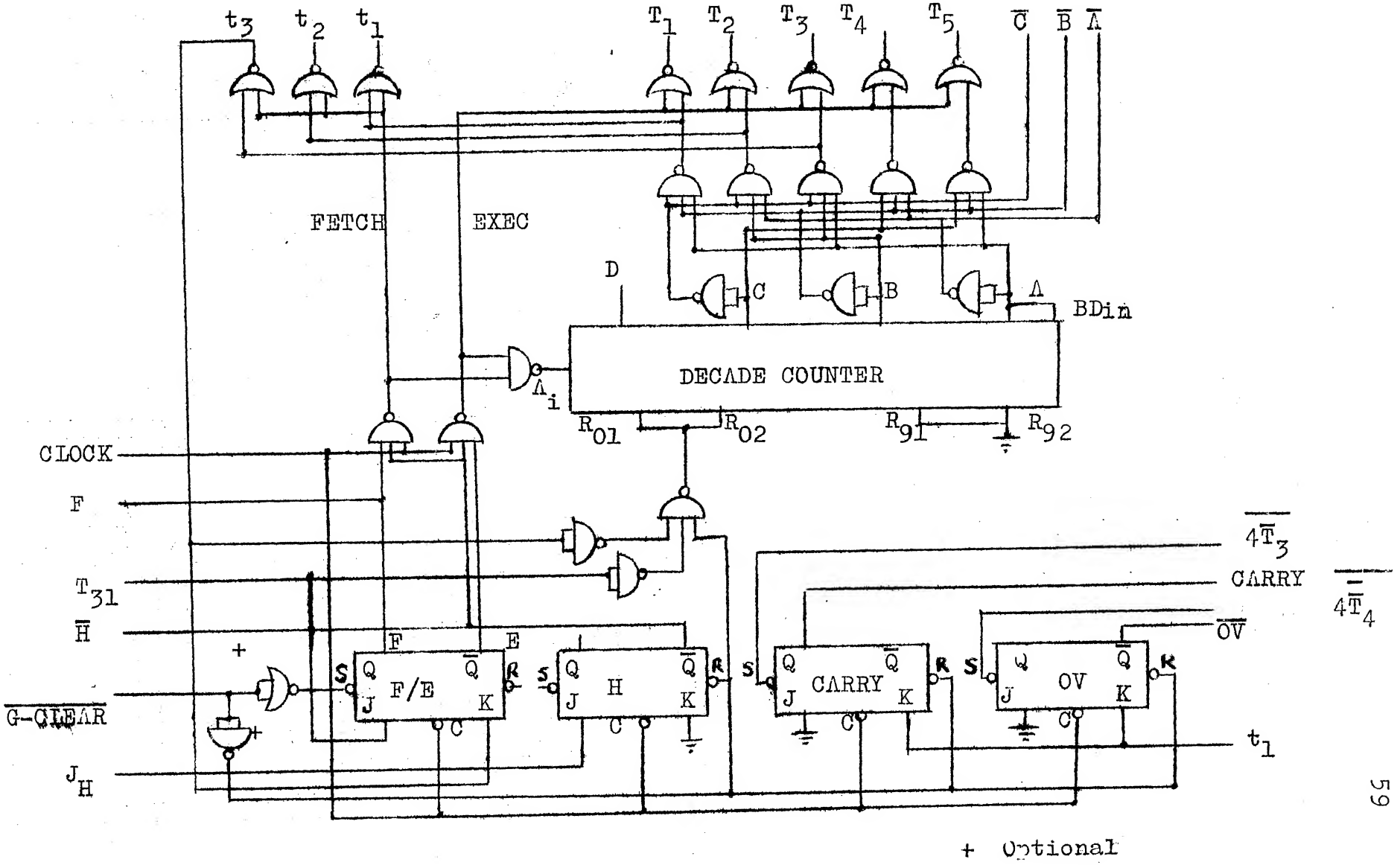


Fig. 3.10 COUNTER and DECODER

(d) OVERFLOW FLIP-FLOP (OV): This FLIP-FLOP gives a visual indication whenever an overflow occurs during the addition operation. Once this FLIP-FLOP is set, the next clock pulse sets the Halt FLIP-FLOP thus bringing the computer to the Halt state. Like the carry FLIP-FLOP, overflow FLIP-FLOP is also reset in each Fetch cycle and also by the G-clear pulse.

3.7.3 Instruction register:

As previously mentioned in chapter 2 the first 3 bits of this register forms the OP code part and the remaining five bits forms the address part. The mode control input is controlled by 't_2' pulse and CLOCK 2 input by the 'CLOCK' signal.

3.7.4 OP code decoder:

This decodes the operation code part of the instruction register. Figure 3.11 shows the instruction register together with the OP code decoder and P-register.

3.7.5 P-Register:

This 5 bit register selects the address of memory location in the Fetch state. This serves as a counter for

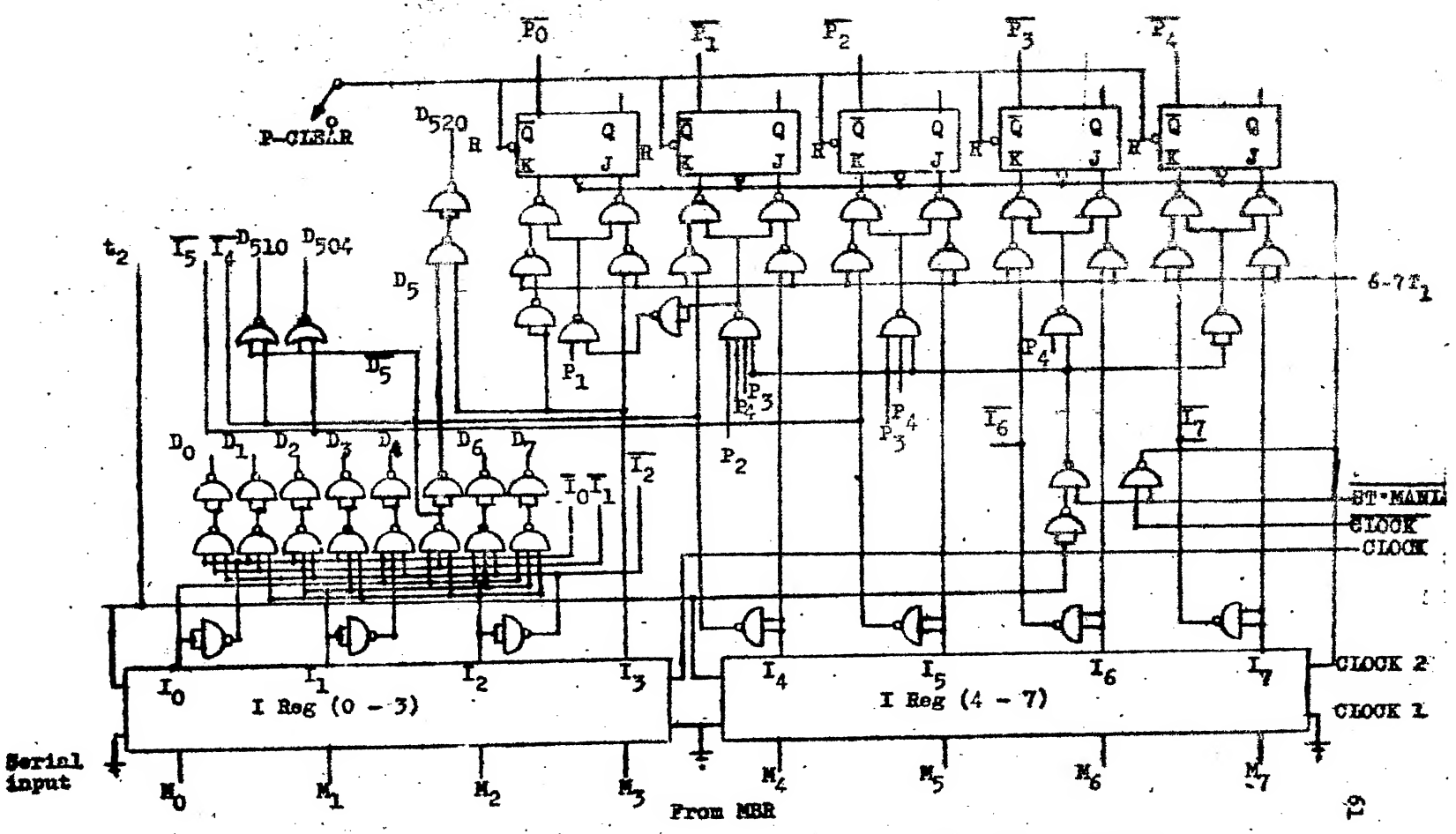


Fig. 3.11 I and P registers with OP Code decoder

't_2' pulses as well as the CTP P pulses applied manually by the INC-P microswitch. Information is transferred to this from the I-register during Branch type instructions.

3.7.6 CLOCK pulse generator:

A basic clock of 45 KHz is constructed out of NOR gates. These clock pulses are used both in 'Auto' and 'Manual' modes. In 'Auto' mode, the clock signal is gated by a NOR latch. This NOR latch is controlled by the 'G-clear', button as well as the 'START' button. The 'G-clear' button inhibits the clock signal in 'Auto' mode and 'START' button initiates it.

The asynchronous 'INC-P' and 'STEP' inputs are synchronized with the clock pulse by a synchronising circuit consisting of two FLIP-FLOPS. A single pole 3 way switch selects any one of these modes. Figure 3.12 shows the clock generator and the synchronising circuitry.

3.7.7 Pulse distributor:

This portion of the circuitry is responsible for providing various timing pulses for the operation of different microinstructions.

The logical equations for the different timing signals are derived from Table 3.2. Figure 3.13 shows the

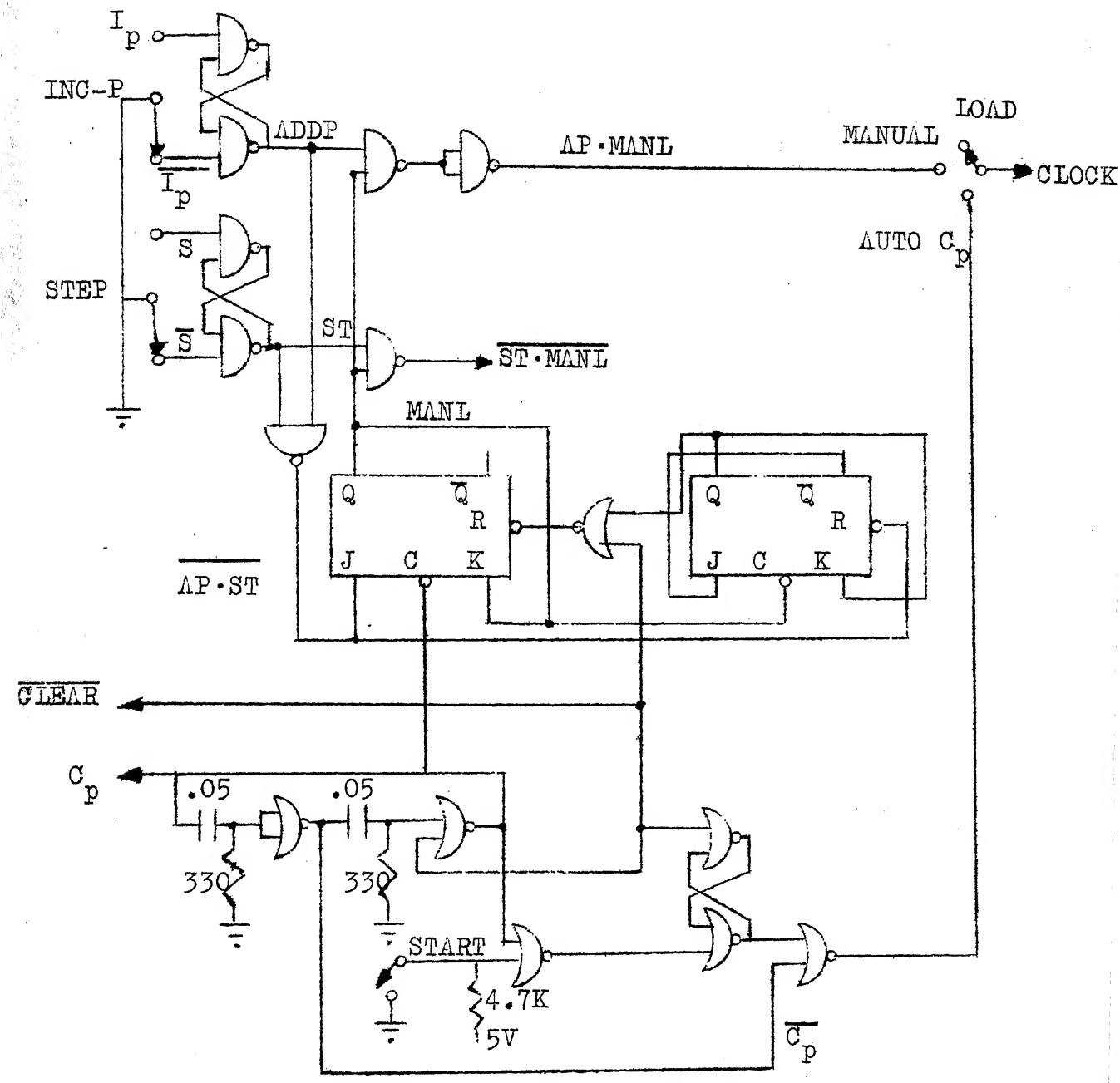


Fig. 3.12 CLOCK pulse generator

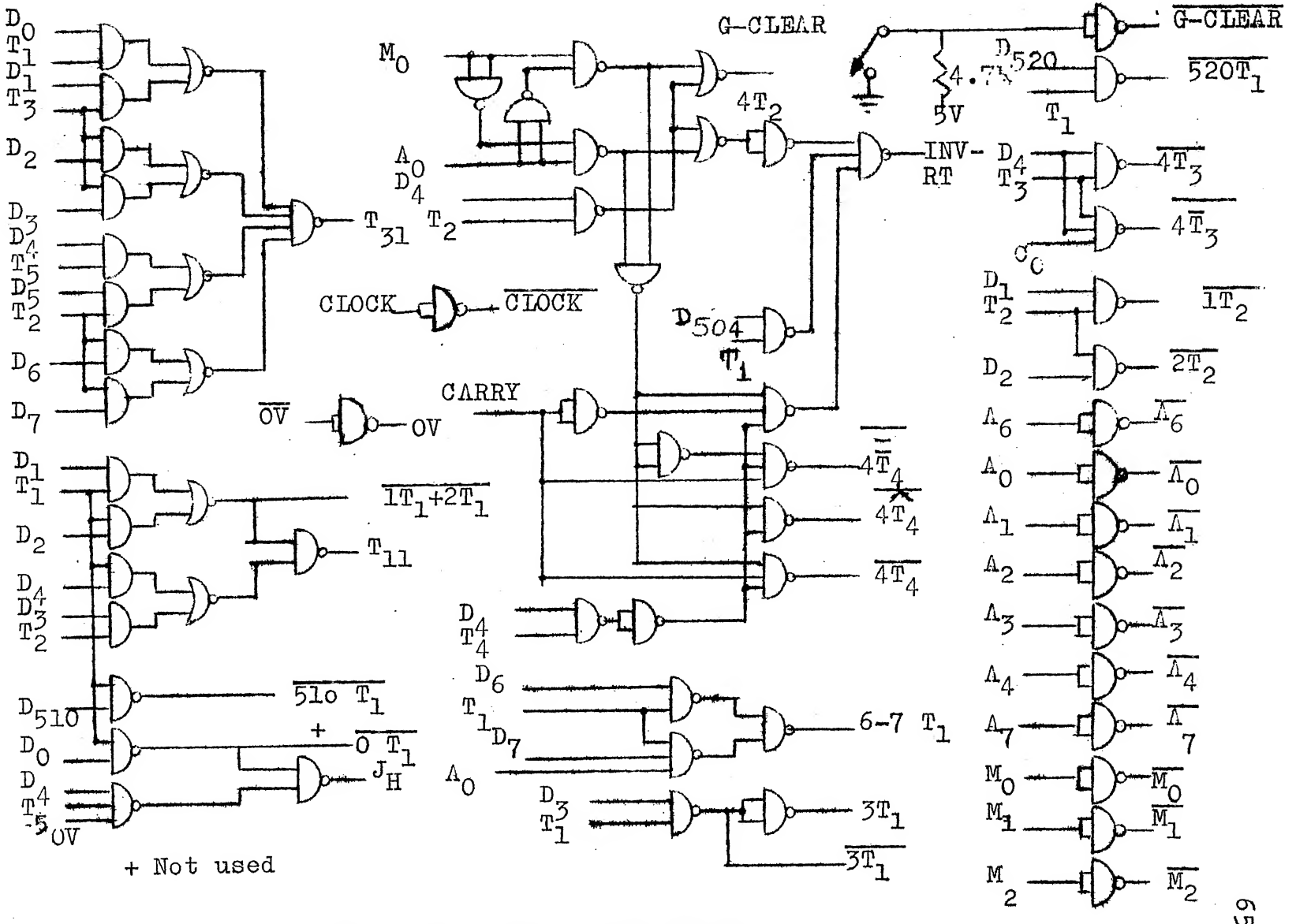


Fig. 3.13 Pulse distributor

logical realization of these timing signals using the different gates.

(i) Halt pulse

$$J_H = 0T_1 + 4T_5$$

$$= D_0 \cdot T_1 + D_4 \cdot T_5 \cdot 0V$$

$$= \overline{\overline{D_0 \cdot T_1} \cdot \overline{D_4 \cdot T_5 \cdot 0V}}$$

(ii) Memory select/during execution phase pulse

$$T_{11} = 1T_1 + 2T_1 + 3T_2 + 4T_1$$

$$= D_1 \cdot T_1 + D_2 \cdot T_1 + D_3 \cdot T_2 + D_4 \cdot T_1$$

$$= \overline{\overline{D_1 \cdot T_1 + D_2 \cdot T_1} \cdot \overline{D_3 \cdot T_2 + D_4 \cdot T_1}}$$

(iii) End of Execution

$$T_{31} = 0T_1 + 1T_3 + 2T_3 + 3T_3 + 4T_5 + 5T_2 + 6T_2 + 7T_2$$

$$= D_0 \cdot T_1 + D_1 \cdot T_3 + D_2 \cdot T_3 + D_3 \cdot T_3 + D_4 \cdot T_5 +$$

$$D_5 \cdot T_2 + D_6 \cdot T_2 + D_7 \cdot T_2$$

$$= \overline{\overline{D_0 \cdot T_1 + D_1 \cdot T_3} \cdot \overline{D_2 \cdot T_3 + D_3 \cdot T_3} \cdot \overline{D_4 \cdot T_5 + D_5 \cdot T_2} \cdot \overline{D_6 \cdot T_2 + D_7 \cdot T_2}}$$

(iv) INVRT pulse to Accumulator

$$INVRT = 504\ T_1 + 4\ \overline{T_2} + 4\ \overline{T_4}$$

$$= D_{504} \cdot T_1 + D_4 \cdot T_2 \cdot A_0 \cdot \overline{M_0} + D_4 \cdot T_4 \cdot (M_0 \oplus A_0) \cdot \overline{CARRY}$$

$$= \overline{\overline{D_{504} \cdot T_1} \cdot (\overline{D_4 \cdot T_2} + \overline{A_0 \cdot \overline{M_0}}) \cdot \overline{(D_4 \cdot T_4)} \cdot (M_0 \oplus A_0) \cdot \overline{CARRY}}$$

(v) Address transfer pulse to P-register:

$$6\text{-}7\ T_1 = 6\ T_1 + 7\ T_1$$

$$= D_6 \cdot T_1 + D_7 \cdot A_0 \cdot T_1$$

$$= \overline{\overline{D_6 \cdot T_1} \cdot \overline{D_7 \cdot A_0 \cdot T_1}}$$

A few more timing pulses are also needed. The logic equations for these signals are too trivial to be included here. The circuitry for all these signals are also shown in Figure 3.13.

A few gates are used as inverters to drive the LED's and also for feeding to some other gates. The inputs to these inverters are A_0, A_1, A_2, A_3, A_4, A_7, M_0, M_1, M_2, A_6, CLOCK, $\overline{OV}$, and G-clear.

CHAPTER IV

HARDWARE DETAILS AND MAINTENANCE

This chapter deals with the fabrication of the unit and also discusses some of the possible faults which may occur at a later time. A systematic approach for the detection and correction of any fault occurring in the system is given at the end of the chapter.

4.1 Printed circuit board specifications:

Once the logic design was completed the circuitry was split and arranged properly to fit into a printed circuit board (PCB) having a 22 pin edge-connector. To give maximum packing density using this 22 pin edge-connector a 20 cm x 17.5 cm double sided board was kept as standard. An offset of 0.85 cm from the center provided in each PCB avoids the danger of reversing the card during insertion. The connections between the two sides of the card were done using jumper wires. A total of 16 PCBs were used in the unit including the two boards used for connecting LED's.

4.2 Circuit schematic

The different cards were packed properly with IC chips to utilize all the terminals efficiently and economically Most of the different functional units were arranged to go

into different PCBs. While labelling the PCBs care has been taken to see that these names indicate their functional performance as far as possible.

The different PCBs together with their functional chips are given in Table 4.1.

Table 4.1 FUNCTIONS OF DIFFERENT PCBs

Serial Number	Name of PCB	Functional chips included
1.	MMY	Memory chips, two 4 bit shift registers used as MBR, and associated logic circuitry.
2.	DATA	NAND gates for the insertion of data into MBR from DATA switches.
3.	INV 2	NAND and NOR gates mostly used for inverting purposes.
4.	INV 1	NAND gates used as inverters and also as latches for 'INC-P' and 'STEP' switches.
5.	ACC 2	4 bit shift register used as ACC (4-7) bits, one 4 bit full adder and associated logic circuitry for the accumulator register.
6.	ACC 1	4 bit shift register used as ACC (0-3) bits, one 4 bit full adder and associated logic circuitry for the accumulator register.

Serial	Name of PCB	Functional chips included
7.	CHIPS	NAND and NOR gates used for generating the memory control signals CS and WR.
8.	I-P REG	4 bit shift register used as I-Register (4-7) bits, FLIP-FLOPS for P-Register and associated logic circuitry using NAND gates.
9.	CLOCK	FLIP-FLOPs and NOR gates for generating clock pulses and NAND and AND-OR-INVERT gates for memory address selection.
10.	DTR 3	NAND gates for generating the clock pulses for MBR and also for the generation of timing pulses. (Distributor of timing pulses).
11.	DTR 2	NAND and NOR gates for timing pulses (Distributor of timing pulses).
12.	DEC	4 bit shift register used as I-Reg (0-4), and NAND gates to form the OP code decoder (DECODER).
13.	DTR 1	NAND gates and AND-OR-INVERT gates to generate the end of execution signal (T_{31}) and a few other timing signals. (Distributor of timing pulses).

Serial Number	Name of OF PCB	Functional chips included
14.	CTR	A decade counter for producing the timing pulses for fetch and execution sequences and the control FLIP-FLOPs.

Power supply terminals were standardized in all the cards.

4.3 Arrangement of cards:

Maximum care has been taken to minimize the length of connecting wires by positioning the PCBs properly. A back panel terminal identification is given in Appendix - 2. Identical labels imply a direct short between those terminals. All these terminal labels were also marked in the appropriate circuits developed in Chapter 3. One extra lead comes out from ACC 1. The 'YMM' connector numbered 15 specifies the second 22 pin edge connector used in the 'MMY' card.

.4 Power supply:

The system needs only one power supply unit to supply current of 2 amperes at 5 V dc (regulated to ± 5%). A regulated voltage source was used for this purpose. A fuse rated at 2.5 amperes was used in series with the d.c. output voltage.

Although not shown on the logic diagram, each of the 4 PCBs was bypassed with a 0.01 μF polyster capacitor irectly at the V_{cc} and ground pins. These bypass capacitors re required to suppress noise caused by current spikes gene- ated during switching.

.5 Fault detection and correction:

One of the most important problems in fault-tolerant mputing and computer maintenance is to detect whether there ist faults in various components of a computing system. ult detection usually is required to be completed as quick- as possible so that proper actions, such as reconfiguration, ult location, and repair, can be initiated immediately.

5.1 System failures:

The most likely source of failure in digital systems is e to the stuck at '0' and stuck at '1' faults. Certain ip flops may also cause trouble by not responding to their puts. Loose contact as well as broken leads in wiring troduces failures in the system. Power supply voltage flu- ation beyond the allowable range (4.75 to 5.25 Volts) may so affect the working of the system. The single pole three y make-break mode selection switch may also result in lfunctioning of the system if its contacts are not made perly.

4.5.2 Identification and correction:

Once we identify that some fault exists in the system, though the exact identification to the chip level, is a tedious process, its correction is very easy.

Whenever a fault is suspected operate the computer in step mode and note down the contents of all registers and FLIP-FLOPs at each step of operation. From an examination of these, we can more or less identify the faulty card.

To facilitate the test procedure some important test points are brought out on the front panel. These include C_p, CLOCK, CS_0, CS_1, WE, CARRY and power supply terminals.

The 'G-CLEAR' button initiates the clock signal. This signal can be tested at the 'C_p' terminal which indicates any fault in the clock generator.

Clock signal will be present at the 'CLOCK' terminal when the machine is running in 'Auto' mode. This tests for any fault in the latch provided after the clock generator. In 'Manual' mode a single clock pulse is produced at the 'CLOCK' terminal whenever the 'STEP' button is pressed. This checks for any faults in the synchronisation circuit used for synchronising the asynchronous input.

'CS_0' and 'CS_1', remains at the logic '1' level during the normal mode of operation. Only during READ/WRITE operations this is brought to logical '0' level. Similarly 'WE' remains at logic '1' level in the normal mode. 'WRITE' operation brings it to logic '0' level. These control signals indicate any malfunctioning present in the memory chips.

The setting of the 'CARRY' FLIP-FLOP occurs only during the third pulse ($4\ \overline{T}_3$) in the Addition operation on receiving an end around carry. This helps to find out any fault in the 4 bit full adder chips used in arithmetic operations.

All the above mentioned test points give only an overall view of the possible failures in the system. For an exact identification of the fault, the suspected card must be taken out and tested for its operation.

An extra female connector with all its leads connected to a printed circuit board is provided for this purpose. The suspected card must be transferred to this extra 22 pin connector and the extra PCB must be inserted in the original connector. The wiring of the system is unaffected by this but at the same time any internal points of the suspected card can be checked using an oscilloscope. Different input combinations can be applied to the test gates by changing the

program stored in the computer. For each of these combination the outputs can be verified. This helps to pin-point the faulty chip.

In certain cases the stuck at '0' or stuck at '1' fault occurs due to a short between the outputs of two gates. This must be verified and corrected by tracing the wiring on the panel. Once it is identified that the fault is due to a gate, the only option is to replace the faulty chip by a new one.

For a good understanding of the test procedure mentioned above an example is given here.

Fault: A program and its result is given. The program is stored from memory cells 0 through 5. Result is stored in address 6.

0	1	04
1	2	05
2	3	06
3	0	00
4	1	33
5	5	14
6		

Result in address 6 after Computation is 727.

The correct result of this problem is 627.

Identification procedure:

1. Run the computer in 'MANUAL' mode starting from address '0'. Let us assume that the computer functions properly till the contents of address 5 is brought to MBR in the second instruction by the $2T_1$ pulse. The next pulse ($2T_2$) produces a wrong result in accumulator. The contents of I-Reg, accumulator and MBR after the execution of the two pulses $2T_1$ and $2T_2$ are given below.

	After $2T_1$ pulse								After $2T_2$ pulse							
I	0	1	0	0	0	1	0	1	0	1	0	0	0	1	0	1
A	0	0	1	1	1	0	1	1	1	1	(1)	1	0	1	1	1
MBR	1	0	1	0	1	1	0	0	1	0	1	0	1	1	0	0

By comparing the two sets it is clear that a fault had occurred in the 2nd bit of accumulator. This fault can occur in two ways:

(i) The NAND output of the gate connected to the 2nd bit of accumulator must be remaining at '1' level (Stuck at '1').

(ii) NAND output of the gate is remaining at '0' level but the input is not getting transferred to accumulator. Both of these conditions may occur due to loose connection also. To verify this testing must be carried out at the IC terminals itself.

The solution for the first case is to replace the NAND gate by a new one. In the second case replace the accumulator (0-3) chip.

Similar test procedures can be adopted for detecting any fault in the system.

Some of the common faults and their test procedures are given in Table 4.1.

Table 4.1 : Tests for detecting faults in DEMOC

Sl. No.	Fault	Test
1.	Contents of memory changes	Check the control terminals CS and WE of the memory chips.
2.	Some of the inputs from DATA switches are not entered into MBR	Hold time of these inputs may not be sufficient. Connect a capacitor at the MBR input terminals.
3.	Computer not running in 'Auto' mode	Check for 'CLOCK' pulse
4.	Accumulator (0-3)/ (4-7) bits changes arbitrarily	Check the clock 2 input terminal.

For ease of maintenance the base diagrams of all the IC chips used are given in Appendix - 3.

CHAPTER V

CONCLUSION

5.1 Biography of the machine:

The present work is a simplified version of SMAC. The design originally proposed centered around a bigger machine of 64 words memory capacity of 12 bits each, 4 G.P.R's, and 16 OP codes. All arithmetic instructions as well as input-output instructions were included in the design. Input output instructions were intended to solve complex problems by splitting them into smaller one's and inputting them by interrupting the machine. But the cost and complexity of such a machine did not allow its fabrication. A simple, low-cost demonstration computer like the one constructed in this project is easy to fabricate and at the same time it does familiarize a beginner with the essentials of computer hardware. This was the main motivation behind this work.

5.2 Capabilities of the present machine and possible uses:

The logical operations (NAND and NOT) provided in the machine help it to simulate any logical operation using gates. The shift left, shift right, Add and Branch instructions serve as a powerful tool for developing most of the programs

like multiplication, division, squaring etc. The 'store' instruction can be used to store partial results as well as the end results into memory.

The sample programs given in Chapter 2 spell out only a few of the possible uses to which it can be put. The user can develop and try a number of other programs thus exploiting the machine to its full extent.

5.3 Suggestions for further development:

The memory capacity is a limiting factor in solving a number of problems. This can be overcome either by adding 'Input' instruction to the machine or by augmenting the memory by adding additional memory modules. The latter necessitates the need for more number of bits for address selection. Increasing the word length for this purpose becomes a tedious process. Hence, double word instructions will be the best choice for this purpose. By this means the memory capacity can be increased to 256 locations. But as the cost of these chips stands now, the above mentioned process may double the system cost.

APPENDIX 1

ISP (Instruction - Set Processor)

P_c State

A <0:7>	Accumulator
MBR <0:7>	Memory Buffer Register
I <0:7>	Instruction register
OV	Overflow indicator
H	Halt indicator
F	Denotes Fetch sequence
E	Denotes Execution sequence

M_p State

$M[0:37_8]$ <S, 1:7>	Memory

Instruction Format

Instruction/I <0:7>

OP <0:2> := I <0:2>	Operation Code
N = I <5>	Not bit
SL = I <4>	Shift left bit

SR = I ⟨3⟩ Shift Right bit

Addr ⟨0:4⟩ = I ⟨3:7⟩ Address part

P ⟨0:4⟩ Address register

Instruction interpretation process

¬H ∧ F → (MBR ⟨0:7⟩ ← M ⟨P⟩ next fetch

instruction ← MBR ⟨0:7⟩, P ← P+1 ; next fetch

instruction ⌴ execution. execute

Instruction set and instruction execution process

Instruction⌴execution: = (

Load and Arithmetic

CLA (:= OP = 001) (A ← M[Addr]); Load accumulator

STO (:= OP = 011) (M[Addr] ← A); Store accumulator

ADD (:= OP = 100) (OV, A ← A+M [Addr]); next Add

((OV=1) → H ← 1); Halt

Logical instructions

NAND (:= OP = 010) → (A ← A ↑ M[Addr]); Logical NAND

NOT (:= OP = 101) → ((N=1) → A ⟨0⟩ ← A ⟨0⟩; Logical NOT

A ⟨1:7⟩ ← ¬A ⟨1:7⟩) ;

Shift instructions

SHL (: = OP = 101)→((SL=1)→A←A*2); Shift left logical

SHR (: = OP = 101)→((SR=1)→A← A/2); Shift right logical

BR (: = OP = 110)→(P← Addr); Unconditional branch

BRM (: = OP = 111)→((A⟨0⟩ = 1)→P←Addr); Branch on negative

Halt

HLT (: = OP = 000)→(F←1; H←1) ; Computer halts.

APPENDIX II

BACK PANEL TERMINAL DIAGRAM

Pin No.	MMY	DATA	INV 2	INV1	ACC2	ACC1	CHIPS	I-P REG
1.	$\overline{M_{11}}$	$\overline{M_{11}}$	$\overline{M_1}$	$\overline{I_p}$	M_6	$\overline{520T_1}$	T_2	$\overline{I_3}$
2.	$\overline{M_{10}}$	$\overline{SW_1}$	M_1	I_p	$\overline{A_3}$	INVRT	D_3	I_3
3.	$\overline{M_{12}}$	$\overline{M_{10}}$	$\overline{M_o}$	S	$\overline{520T_1}$	A_o	WRITE	$\overline{CLOCK}$
4.	M_o	SW_o	M_o	$\overline{S}$	M_7	A_1	C_p	$\overline{I_4}$
5.	M_1	$\overline{M_{12}}$	M_2	ST	M_4	A_2	READ	$\overline{P_o}$
6.	M_3	$\overline{SW_3}$	$\overline{M_2}$	$\overline{A_o}$	A_4	A_4	$\overline{MMY\ READ}$	$\overline{ST \cdot MANL}$
7.	CLOCK2	$\overline{M_{13}}$	CLOCK	A_o	A_5	$\overline{2T_2}$	WE	t_2
8.	M_2	SW_2	$\overline{CLOCK}$	$\overline{AP \cdot ST}$	M_5	M_o	$\overline{P_o}$	M_7
9.	$\overline{M_{13}}$	ENTER MBR	OV	A_1	INVRT	M_1	CS_o	M_6
10.	$\overline{M_{14}}$	$\overline{M_{15}}$	$\overline{OV}$	$\overline{A_1}$	C_{01}	M_2	t_1	M_5
11.	$\overline{M_{15}}$	$\overline{SW_5}$	$\overline{ST \cdot MANL}$	A_2	$\overline{510T_1}$	$\overline{4T_4}$	$\overline{I_3}$	M_4
12.	M_4	$\overline{M_{14}}$	MANL	$\overline{A_2}$	$\overline{1T_2}$	M_3	T_{11}	P-CLEAR
13.	M_7	SW_4	ST	$\overline{A_4}$	$\overline{4T_3}$	C_{01}	CS_1	$\overline{P_1}$
14.	M_6	V	V	V	V	V	V	V
15.	M_5	SW_6	G-CLEAR	A_4	$\overline{4T_4}$	C_o	I_3	$\overline{P_2}$
16.	$\overline{M_{16}}$	$\overline{M_{16}}$	$\overline{G-CLEAR}$	$\overline{A_3}$	$\overline{A_5}$	$\overline{510T_1}$	D_5	$\overline{I_5}$
17.	$\overline{M_{17}}$	SW_7	$\overline{MDE+t_1}$	A_3	$\overline{2T_2}$	A_3	$\overline{I_4}$	$\overline{P_4}$
18.	$\overline{4T_1+READ}$	M_{17}	ADD P	ADD P	$\overline{A_4 \cdot A_5 \cdot A_6 \cdot A_7}$	CLOCK	$\overline{I_5}$	$\overline{I_6}$
19.	$\overline{MDE+t_1}$	A_6	AP·MANL	--	A_6	$\overline{4T_4}$	D_{510}	$6\text{-}7T_1$
20.	$\overline{1T_1+2T_1}$	A_6	t_1	A_7	CLOCK	$\overline{A_4 \cdot A_5 \cdot A_6 \cdot A_7}$	D_{504}	$\overline{I_7}$
21.	$\overline{M_7}$	--	MDE	$\overline{A_7}$	A_7	$\overline{1T_2}$	--	$\overline{P_3}$
22.	$\overline{M_6}$	GND	GND	GND	GND	GND CARRY	GND	GND
	1	2	3	4	5	6	7	8

contd....

APPENDIX II

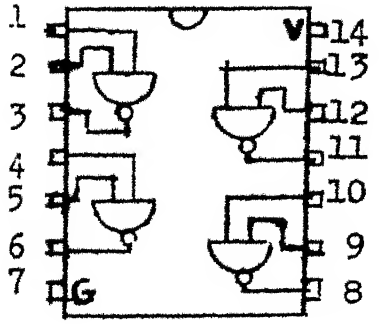
BACK PANEL TERMINAL DIAGRAM

Pin No.	CLOCK	DTR 3	DTR 2	DEC	DTR 1	CTR	YMM
1.	$\overline{F}$	C_o	D_3	t_2	D_2	CLOCK	A_1
2.	$\overline{P}_2$	C_p	$3T_1$	M_3	D_1	F	A_o
3.	F	ENTER MBR	6-7 T_1	M_2	T_3	$\overline{H}$	CS_1
4.	$\overline{I}_5$	$\overline{\text{ENTER}}$	D_6	M_1	D_3	$\overline{A}$	CS_o
5.	$\overline{I}_4$	CLOCK 2	M_o	M_o	D_o	$\overline{4\ T_3}$	a_o
6.	$\overline{P}_1$	$\overline{\text{CLOCK}}$	D_{504}	D_o	T_1	$\overline{4\ T_4}$	WE
7.	START	$\overline{4T_1+\text{READ}}$	$4\ T_2$	I_3	T_{31}	J_H	a_3
8.	C_p	T_1	T_2	CLOCK	D_6	$\overline{C}$	a_2
9.	G-CLEAR	D_{520}	D_4	D_3	D_5	$\overline{\text{G-CLEAR}}$	a_1
10.	a_o	$\overline{520T_1}$	$\overline{4T_4}$	D_1	T_2	T_5	$3T_1$
11.	a_1	$\overline{4T_3}$	T_4	D_4	D_7	T_4	$4T_2$
12.	a_3	$\overline{4\ T_3}$	CARRY	D_2	D_4	T_{31}	A_2
13.	a_2	READ	$\overline{4\ T_4}$	D_6	T_5	$\overline{OV}$	A_3
14.	V	V	V	V	V	V	V
15.	AUTO C_p	D_4	$\overline{4T_4}$	D_5	OV	$\overline{B}$	A_6
16.	$\overline{\text{AP·ST}}$	T_3	MDE	D_{520}	J_H	T_3	A_4
17.	$\overline{P}_4$	D_1	$\overline{\text{ENTER}}$	$\overline{D_5}$	$\overline{OT_1}$	T_1	A_5
18.	$\overline{I}_7$	T_2	A_o	$\overline{I}_1$	D_{510}	T_2	A_7
19.	$\overline{P}_3$	$\overline{1T_2}$	T_1	D_7	$\overline{510T_1}$	CARRY	$\overline{M}_2$
20.	$\overline{I}_6$	D_2	D_7	$\overline{I}_o$	T_{11}	t_1	$\overline{M}_3$
21.	MANL	$\overline{2T_2}$	INVRT	$\overline{I}_2$	$\overline{1T_1+2T_1}$	t_2	$\overline{M}_4$
22.	GND	GND	GND	GND	GND	GND	GND
	9	10	11	12	13	14	15

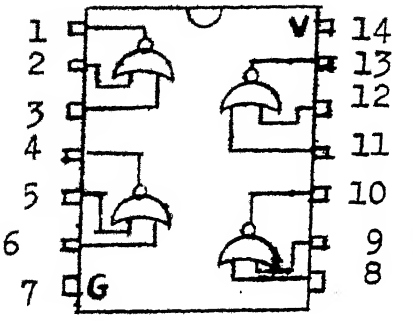
APPENDIX III

IC BASE DIAGRAMS

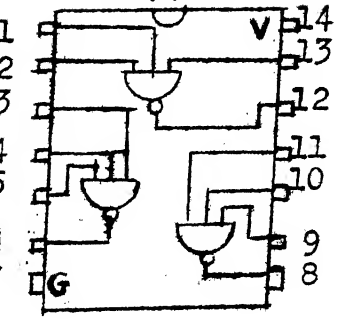
SN7400



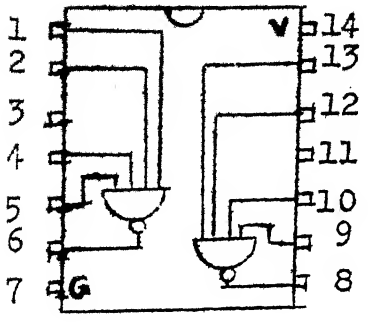
SN7402



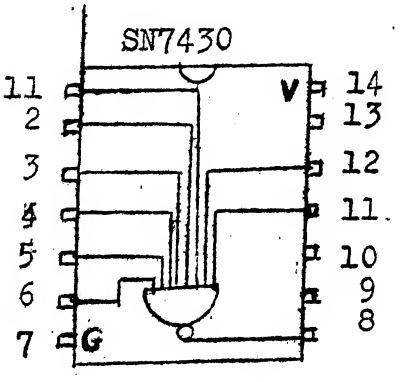
SN7410



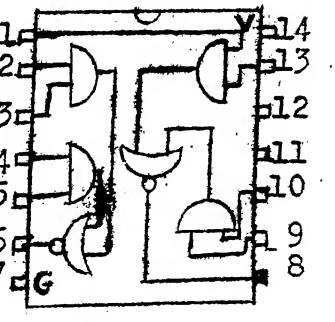
SN7420



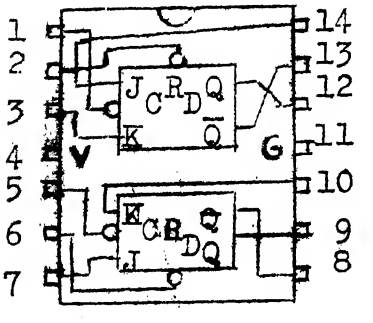
SN7430



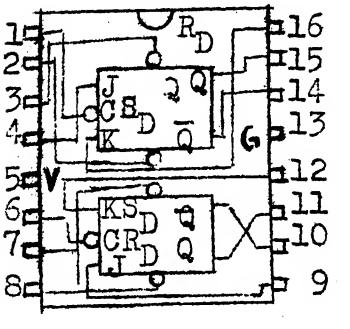
SN7451



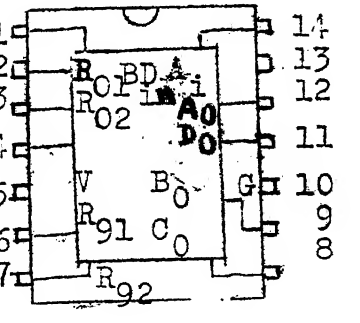
SN7473



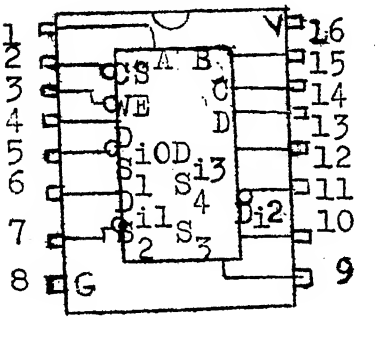
SN7476



SN7490



IM5501

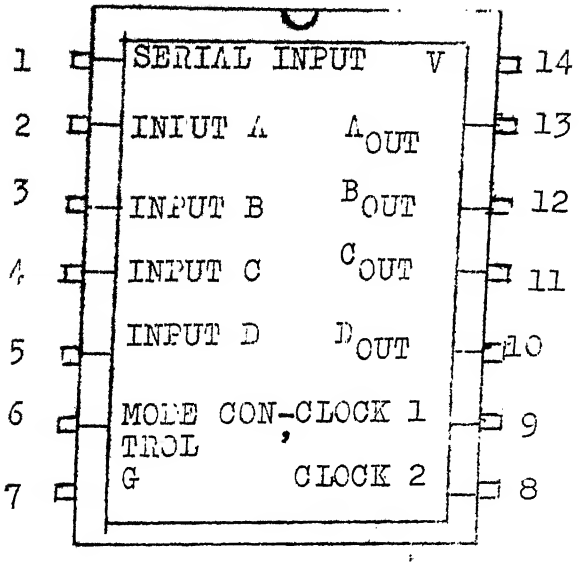


S_1, S_2, S_3, S_4 - O/P

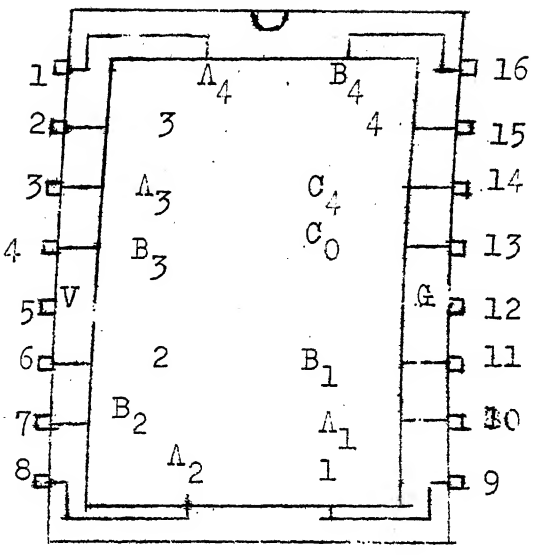
A, B, C, D - Address

$D_{i0}, D_{i1}, D_{i2}, D_{i3}$ - Inputs

SN7495



SN7483



REFERENCES

1. Bartee, T.C., I.L. Lebow, and I.S. Reed, Theory and Design of Digital Machines, McGraw-Hill, 1962.

2. Bell, G. and A. Newell, Computer Structures: Readings and Examples, McGraw-Hill, 1971.

3. Chu, Y., Digital Computer Design Fundamentals, McGraw-Hill, 1962.

4. Desautels, E.J., On Computing facilities for Computer Science, Computer, Vol. 7, No. 11, November 1974, pp. 39-49.

5. Hamblen, J.W., Using Computers in Higher Education, CACM, Vol. 14, No. 11, November 1971, pp. 709-712.

6. Hill, F.J. and G.R. Peterson, Digital Systems: Hardware Organization and Design, Wiley, 1973.

7. Kautz, W.H., Fault Testing and Diagnosis in Combinational Digital Circuits, IEEE Transactions on Computers, Vol. C-17, April 1968, pp. 352-366.

8. Morris, R.L. and J.R. Miller, Designing with TTL Integrated Circuits, McGraw-Hill, 1971.

9. Rajaraman, V. and T. Radhakrishnan, Logic Design of Digital Computers and Systems, (to be published).

10. Richards, R.K., Digital Design, Wiley, 1971.

11. Scott, N.R., Electronic Computer Technology, McGraw-Hill, 1970.

12. Sobel, H.S., Introduction to Digital Computer Design, Addison-Wesley, 1970.

13. Susskind, A.K., Diagnostics for Logic Networks, IEEE Spectrum, Vol. 10, No. 10, October 1973, pp. 40-47.

14. Woollons, D.J., Introduction to Digital Computer Design, McGraw-Hill, 1972.

15. Vir Singh, C. and H.N. Mahabala, Design of an Educational Digital Computer and General Purpose Logic Simulation Program, Technical Report, Computer Centre, IIT Kanpur, 1968.

S-M-RAD-DEM